blender art MAGAZINE

blender learning made easy

Blender Materials

- Artistic Glow Using Blender's Compositor Nodes
- DOF Using Blender's Compositor Nodes
- Creating A Realistic Environment for BGE
- Blender and Displacement Mapping
- Blender and Vector Blur

Thank you all !

Sandra Gilbert
Managing Editor

Welcome to our 1st Anniversary Issue! Yes, that s right, we actually made it a whole year! When Gaurav and I started this magazine, we had no idea how much we would end up needing to learn. But learn we did, as well as adding staff members to help out with proofing and website maintenance.

It has been an eventful year for both BlenderArt Magazine as well as the Blender community itself. We have seen the creation and release of Elephant s Dream, the addition of many needed and just plain wanted features in Blender. The Blender community continues to grow at a phenomenal rate. BlenderArt Magazine reader numbers have continued to grow as well. With each new issue, we gain a larger audience. And we appreciate all of you!

We have had an amazing response from the community in terms of contributions to the magazine. While Gaurav and I could have still put out a magazine without all their help, it would not have been anywhere near as interesting. I d like to recognize all of our contributors at this point; they deserve a round of applause for their efforts on all our behalves.

Here they are, in no particular order, and I sure hope I didn t forget anyone; if I did I apologize in advance.

Enrico Valenza	Christian	Martin Mackinlay
Juan J. Pena	Guckelsberger	Ed Cicka
David Lettier	Manuel Perez	Cory King
Zsolt Stefan	Claudio Andaur	Early Ehlinger
Claudio Malefico	JosÈ Mauricio	Mariano Hidalgo
Andaur	Rodas R.	Edouard de Mahieu
Stefano Selleri	RogÈrio Perdiz	Sergey Prokhorchuk
Claas Eicke Kuhnen	Manuel Bastioni	Roja
Daniel LaBarge	Alessandro Proglio	Andreia Leal
Christopher Kulla	Antonio Di Cecca	Schemid & Zag
Kernon Dillon	Giovanni Lanza	Diego Restrepo ParÌs

Okay, now on to what you really want to know about - what s in this issue! Materials, lots and lots of material information. Admit it, you want it and you need it. You put enormous effort into your models and you know, as well as I do, that you just have to add materials at some point. With the addition of Nodes, or more commonly called Noodles, whole new effects and materials are possible. First, we'll answer a simple question What are Materials? Then, we have several Node tutorials, as well as a look at Vector Blur and Ramp Shading. We also take a look at some current Educational Resources available for Blender users.

So here we go, let s get our Blender world all colored up!

Happy Blending
sandra@blenderart.org

blenderart MAGAZINE

EDITOR /DESIGNER
Gaurav Nawani
gaurav@blenderart.org

MANAGING EDITOR
Sandra Gilbert
sandra@blenderart.org

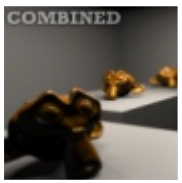
WEBSITE
Nam Pham
nam@blenderart.org

PROOFER
Kernon Dillon
Phillip A. Ryals
Kevin C. Braun
Derek Marsh

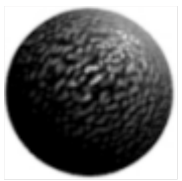
WRITERS
Olivier Saraja
Daniel LeBarge
John Allie
Michael Wach

COVER ART
Wise Men
by Mathias Pedersen

Artistic Glow Using Blender's Compositor Nodes 6



DOF Using Blender's Compositor Nodes 8



Creating Realistic Environment for BGE 11

Blender And Displacement Mapping 24

Blender and Vector Blur 27

Blender Materials

With the multitude of tools available in Blender, just about any material can be convincingly created. But there is a saying in CG, it's not the tool, it's the artist that creates art. The same can be said about materials. All the latest, new tools aren't going to create realistic or amazing materials for you. You still have to know how to use them and more importantly, you have to know how materials are created and what makes them convincing.

So, instead of focusing on tools, we are going to take a look at what materials are and what makes a good material. With that information, you can then use all the amazing tools more effectively. Let's start with a few definitions.

- **Material:** is the base substance of a surface (i.e. wood, metal, glass)

- **Texture:** is the adjective of the material (i.e. rusty iron, brushed steel, soiled cloth, frosted glass)

Anytime you create a new material, you should gather reference materials. It can be an object with the material you are trying to create or a good photograph of it. These reference materials will help you "visualize" what you are trying to create. When gathering reference, very rarely will you find pristine materials without signs of wear and tear in real life. There will be dust, rust, scratches, dents and a multitude of other signs that this material has gone through over the period of time. Notice these signs of wear and decide if they are signs of "material" or "texture".

Texture attributes/qualities can be broken down by answering the following questions:

What are you?
Often the simple act of identifying an object can help you understand what kind of materials you will be creating.
Example: a fire hydrant tells you its shape; that it is made of thick steel left out in the weather, and most often painted.

What is your essence? Try to identify the most important feature of the material
Example: a radiator's most important feature might be its rusty quality. As you identify the most important feature, keep looking deeper and noticing each unique feature of the material, it will give you a list of features to add to the material; adding more realism and depth with every one you add.

What are you made of? Before dissecting the many layers of textures that make up a material, you need to identify the base material. By identifying the base material, you will more easily see the details (textures) that make it that particular material.
Example: an old metal sign; knowing that it is metal, leads you to realize that some metals are prone to rusting, so look for signs of rust and corrosion. Is the sign painted? (Old paint has a tendency to chip or flake away, new paint will look shiny and somewhat reflective.)

What do you sound like? As odd as it might sound, sometimes your eyes will fool you. Tap on the object, what does it sound like? Metal, glass plastic...

What do you smell like?
Often you can identify important clues by noticing how the object smells. Leather smells different than plastic, different metals have different smells.

How can I see you?
Identify the light source, lighting of the material will play an important part in how it is recreated. Warm light sources would produce warm highlights or hotspots, cool lights the opposite. When gathering your reference materials, make sure to note in a journal or mentally what the light source was, time of day (if outside) so you can recreate the material accurately.

Where are you?
Is the object inside or outside? If outside, it will be exposed to weather, heat, moisture. This will affect its appearance. New objects quickly lose their new appearance by acquiring dents, dings, scratches, dust etc. Knowing where it is located will help identify these qualities.

What do you look like?
Now comes the part where you start describing the texture qualities. The better you can describe what you see, the easier it will be to recreate. Some things to look for:
Color texture
Reflections and shadows
Is the object transparent?
Is the object luminous?
With heat?
Without heat?
From an outside source?
From the object itself?

Luminescent objects can have one or two of these qualities and identified as:
Translucent
Iridescent
Opalescent
Fluorescent
Incandescent

What do you feel like?
How does the object feel? Is it bumpy or smooth, warm or cool, soft or hard?

Tactile qualities:
Temperature
Bumpiness
Roughness
Smoothness

What's your story?
Try to identify the history of the object, how has usage contributed to wear and tear?

By answering these questions, you will better understand your material and how to go about recreating it. It will, of course, take practice and lots of observation to master these skills. You need to learn to train your artistic eye to really see what you are looking at. Soon, breaking materials down will be automatic and your materials will improve dramatically.

The information for this article was paraphrased from Owen Demers book 'Digital Texturing & Painting' (ISBN 0-7357-0918-1). I highly recommend reading this book for anyone wanting to take their material skills to the next level.

Blender News

Ton has announced that Blender is slowly headed for another release. It looks like we may get a new release in time for Christmas. A host of new features have been added. Release logs have already been written for the following features;

Animation
Walk Cycle Modifiers
Proxy Objects, for local control over referenced data from Libraries

Rendering
Render features: tangents and normal maps
Irregular Shadow Buffers
Shadow buffer, Halfway average
Render Baking

Compositing
Matte Nodes
Matte Nodes, tutorial

Links to the following feature documentation will be posted as soon as they are finished.
Modeling
Modifier Stack upgrades
Sculpt modeling, multi-resolution Mesh and Retopo
Fluid Dynamics supporting animated Objects
Multi-level UV editing

Testing builds can be downloaded from www.blenderbuilds.com.

A new blender book has been announced and they are looking for contributors to help with completion of the book. If you are interested in helping, contact Roland Hess at blenderbasics at harkyman dot com.

Editor Roland Hess writes:
"The book can be used as a full introduction to 3D art and Blender, or as a reference for people already experienced with other applications. Each chapter consists of a tutorial section and theory section, meant to speak to new users and give them a grounding in the solid basics needed to succeed with Blender."

Artistic Glow Using Blender's Compositor Nodes

-by Daniel LaBarge

Artistic Glow is an effect that is a component of almost every major photo editing software package. Blender has the ability to recreate this effect, but with more control and more finesse, because of the unique node based system in which glow can be applied. There are many ways of adding glow, including using other programs such as GIMP or Blender's Glow plugin for the sequencer. We are going to take a look at how to add glow using Blender's Compositor Nodes.

Prerequisites

This article assumes that you have a scene which contains some areas of contrast and that you know your way around Blender. It is safe to say that this is for intermediate to advanced users.

Getting Started

To create Glow in Blender we need to do several things. First, enable the Compositor. To do this, please reference Image 1 which shows the Do Composite enabled. You can find these in the Render Buttons (F10) under the Anim Panel.

Now we are ready to begin compositing our scene with Glow. Open up the Node Editor Window and by default it should have a Render Layers node connected to two output nodes, Viewer and Compositor. It is a good idea to open up the UV Image Editor Window and switch the image to Viewer Node to get a larger view of the Viewer output. This helps with fine tuning.

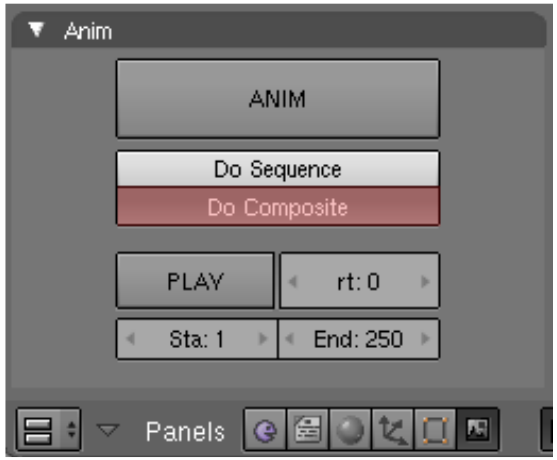


Image 1. Enabling node compositor

Let's first take a look at the image as rendered originally. Obviously the scene is setup with areas of strong contrast to help with applying realistic Specular Glow instead of just Intensity Glow. We will base the glow on this color value, and then mix the glow mask on top to complete the Glow effect.

Creating the Glow Mask

To create the Glow Mask we need to highlight the details in the image. To do this I like to first use a Sharpen Filter. Add a Filter node and hook up the Image input to the Image output of the Render Layers node. Hooking up a Viewer node to the output Image of the Filter node will display the results of the filter effect. We will be using Filter Type Sharpen, as tests have shown it to be the best for enhancing the details of the image, although it creates many artifacts which we will have to deal with later.

Now we need to isolate only the specular areas, or at least the areas with the most intensity, which typically are the areas of specularity. To do this we need to modify the RGB color values. The node for this is the RGB Curves node. Hook the Image output of the Filter node up to the Image input of the RGB Curves node.

Image2. Example scene in different settings

The curve requires tuning according to your scene. You want a picture that exhibits the most contrast between darkness and lightness. To adjust this curve use the Combine curve tab and select locations on the curve and move them. Hooking up a Viewer node to the Image output helps.

After the output has enough contrast you now need to convert this to black and white. For this step we'll need to add an RGB to BW Converter node. Hookup the Image output of the RGB Curves node to the Image input of the RGB to BW Converter node and then hookup the output to a Viewer node to see the results.

We're almost there, but as you can see, the mask is very pixelated and has lots of artifacts. To fix this, we need to blur it.

Add a Blur node, connecting the Value output of the RGB to BW Converter node to the Image input of the Blur node. Use a filter type of Quadratic and a pixel radius (X and Y settings) appropriate for the scene. Enabling Gamma blurring also helps. Hookup a Viewer node to the output to see the final glow mask.

Applying the Glow Mask

To apply the Glow Mask to the Image we need a Mixer node. This node takes the Image inputs and then mixes the two inputs together based on a Filter Type. In most cases you will want to use the Add type. This will perform a pixel-by-pixel addition between the two color values of the Image inputs.

Final Touches

You can now hook up a Viewer node and a Compositor node to the Image output of the Mixer node. You will see the combined images with the Glow effect. If all went correctly, you should now have a picture that has intensity along all of the edges that exhibit specularity or intense light diffusion.

You can use the Factor setting in the Mixer node to control the mixing of the two Image inputs. Fine tuning may be required, and the extra Viewer nodes will help make tuning easy.

Below you will see the output settings for the different values at each stage in the composition. Also included is the schematic for the Glow node setup.

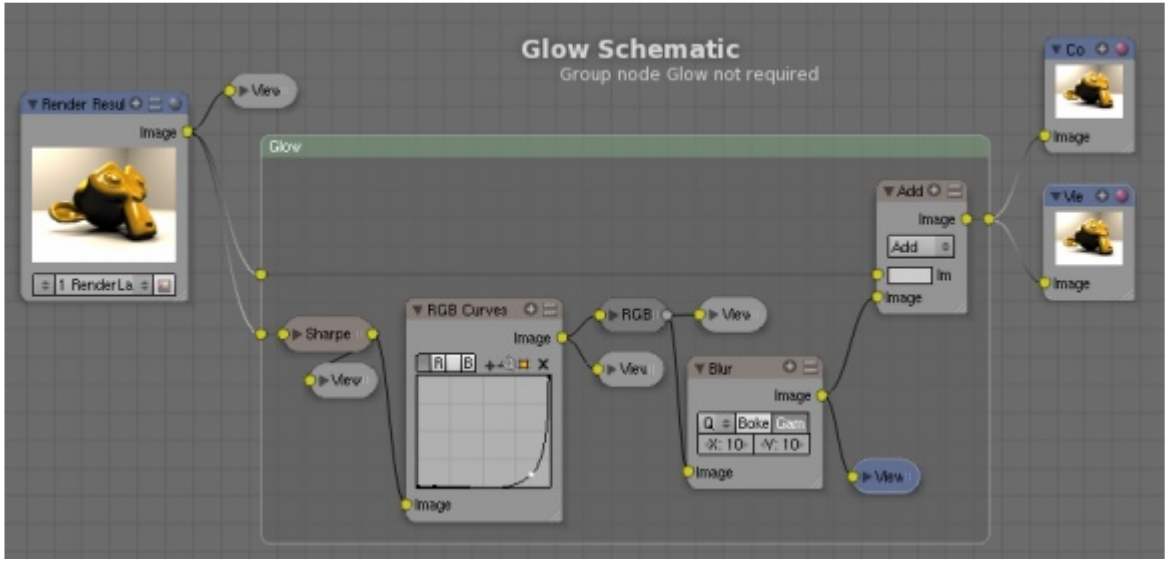


Image3 The Noodles for 'Soft Glow'

NOTE

NOTE: This article was created using a CVS version of Blender and some features shown in screenshots may not be available at the current publication of this article. Through the main Blender release however, the effect described here can be used in the current Blender release.

NOTE

This schematic shows the Glow nodes inside a Group node called Glow. This is not required for the final output.
You should find a reference Blender file included with this issue to track and trace a working example of this setup.

Be sure to visit http://www.idstudios.org for more information on creating Glow and other special effect nodes.

Depth Of Field Using Blender's Compositor Nodes

- by Daniel LaBarge

Depth of Field (DOF) is a very popular effect to be simulated in computer generated imagery as it adds to the overall realism of an image. True, DOF as your eye perceives it is not yet possible with Blender, but a very close simulation of foreground and background blur can be achieved. We will be looking at one method, using Blender's built in Compositor Nodes.

Prerequisites

This article assumes that you have a scene which exhibits a necessity for DOF and that you know your way around Blender. It is safe to say that this is for intermediate to advanced users. .

Getting Started

To create Depth of Field in Blender we need to do several things. First, enable the Compositor and pass the Z Channel (Blender's depth channel) to the Compositor. To do this please reference Image 1 which shows the Do Composite enabled and the Z Pass enabled. You can find these in the Render Buttons (F10) under Anim Panel and Render Layers Panel.

Now we are ready to begin compositing our scene with DOF. Open up the Node Editor Window. By default, it should have a Render Layers node connected to two output nodes, Viewer and Compositor. It is probably a good idea to open up the UV Image Editor Window and switch the image to Viewer Node to get a larger view of the Viewer

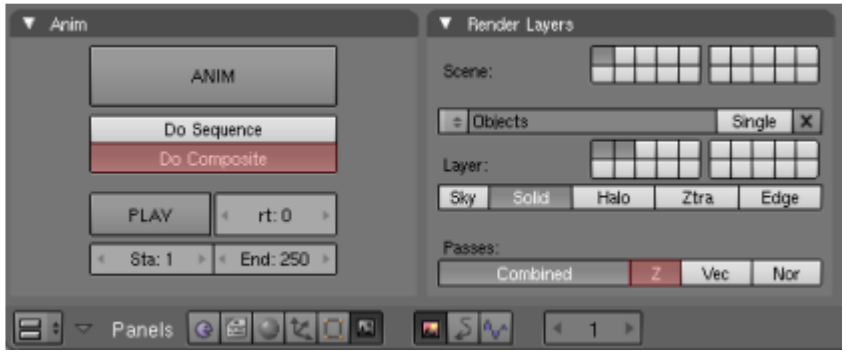


Image 1 Enabling node compositor and Z pass

output. This helps with fine tuning.

Let's take a look at the image as rendered originally. Obviously the scene is setup and the elements are arranged to exhibit good DOF. We will be blurring the color values using the depth channel.

Now let's look at the Z Channel, delivered in the Z Pass to the Compositor. To do this, add a new Viewer and connect the Z input to the Z output of the Render Layer node. In the UV Image Editor click on the icon that shows the Z value. This is what the depth values look like for the scene.

Mapping the Z Channel

These values, although appearing correct, won't actually create a proper DOF effect if sent to the Blur Filter node as the Size input. To fix this, we need another node called Map Value. This node will take the values as inputs and adjust them. The Value input should be connected to the Z output of the Render Layer node. There are several settings, including clamp functions, to set the Minimum value range and the Maximum value range. The two settings we will use are Offset and Size. Offset, in my experience, has always been a negative value and it appears to be the Blender grid units from the camera to the focus. Size appears to be a gradient filter that controls the abruptness of the transition between sharpness and blurriness when used with the Blur node.

Positive value is used for background blur and a negative value is used for foreground blur. The closer the value is to 0, the softer the gradient. Negative values are typically very small, usually no more than -0.250. You can take a look at the new Z channel output by hooking up a Viewer node to the Value output and looking at the Z channel in the UV Image Editor.

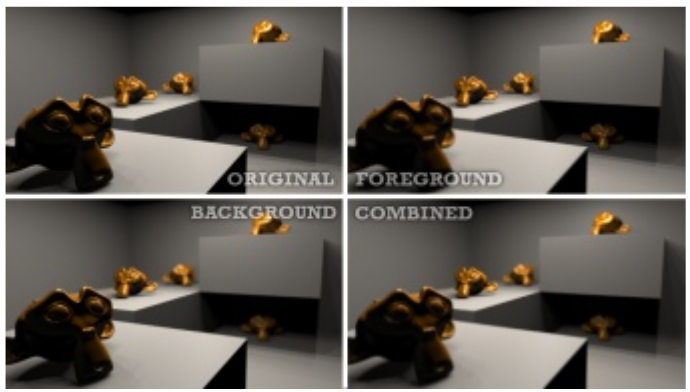


Image2. Example scene in different settings

Create two Map Value nodes and hook them up to the Z output of the Render Layers node. Set Offset to a negative value, approximately the same number of grid units from camera to focus point. The Offset values should be identical or very close. The Size value for one should be a negative value between -0.050 and -0.250, and the other should be a positive value between 0.250 and 1.000.

Blurring the Image

Next we'll need to use these values to blur the image. This is where the Blur node comes into play. Add two Blur nodes and hook up the Image input to the Image output of the Render Layer node. Hook up the Size input to the Value output of the Map Value node. Repeat for the other Map Value node. The Blur node has several settings. I have found that Quadratic filter type and Gamma enabled work best. The X and Y settings control the radius of the blur effect in pixels. For more blur use a higher number. If you hook up a Viewer node to the Image output you will see the results of the DOF and you can edit the Map Values accordingly. The one Viewer node will show the foreground blur and the other will show the background blur.

Now we have the two images blurred but we need the foreground and background blur combined. There are several ways to do this but the proper way is to use another specialty node. This node is called Z Combine. Hook up the Image output of one Blur node to the first Image input of the Z Combine node, and then hook up the other corresponding Map Value's Value output to the Z input of the Z Combine node.

Repeat for the other Blur node and Map Value node. Notice that the corresponding Image input goes to the opposite Value output of the Map Value node. This is important to properly combine the two images without artifacts.

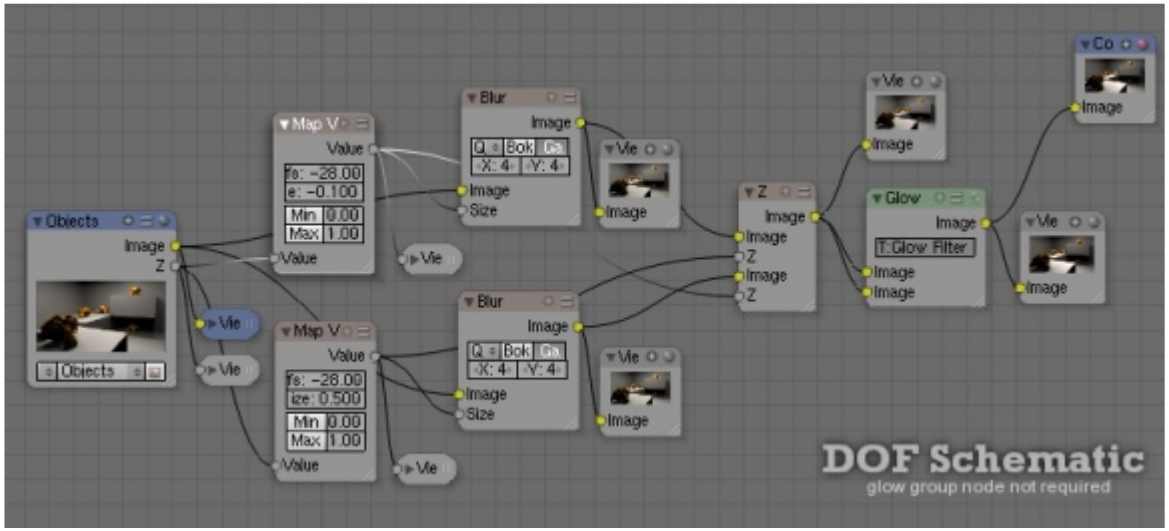


Image3 The Noodles for 'Depth Of Field'

Final Touches

You can now hook up a Viewer node and a Compositor node to the Image output of the Z Combine node and you will see the combined images which will show the DOF effect. If all went correctly you should now have a picture that has foreground blur and background blur with the center range sharp and in focus. Fine tuning may be required, and the extra Viewer nodes will help make tuning easy.

Below you will see the output settings for the different values at each stage in the composition. Also included is the schematic for the DOF node setup.

What About Animations?

Animating DOF with Compositor nodes is a little too lengthy to explain, but the idea hit me one day while trying to mix values. The secret to animated DOF is to mix "keyframed" Map Values along a Time node. As long as you have the DOF set for a particular frame you can control the Mixing using the curve in a Time node. This can get really bulky and I can't wait for the Blender developers to create an input for the Map Value Offset, but until that feature is implemented you can use the reference schematic below.

You should find a reference Blender file included with this issue to track and trace a working example of this setup.

NOTE

This schematic contains a custom Group node for adding DOF to the final composition, and is not required for the final output.

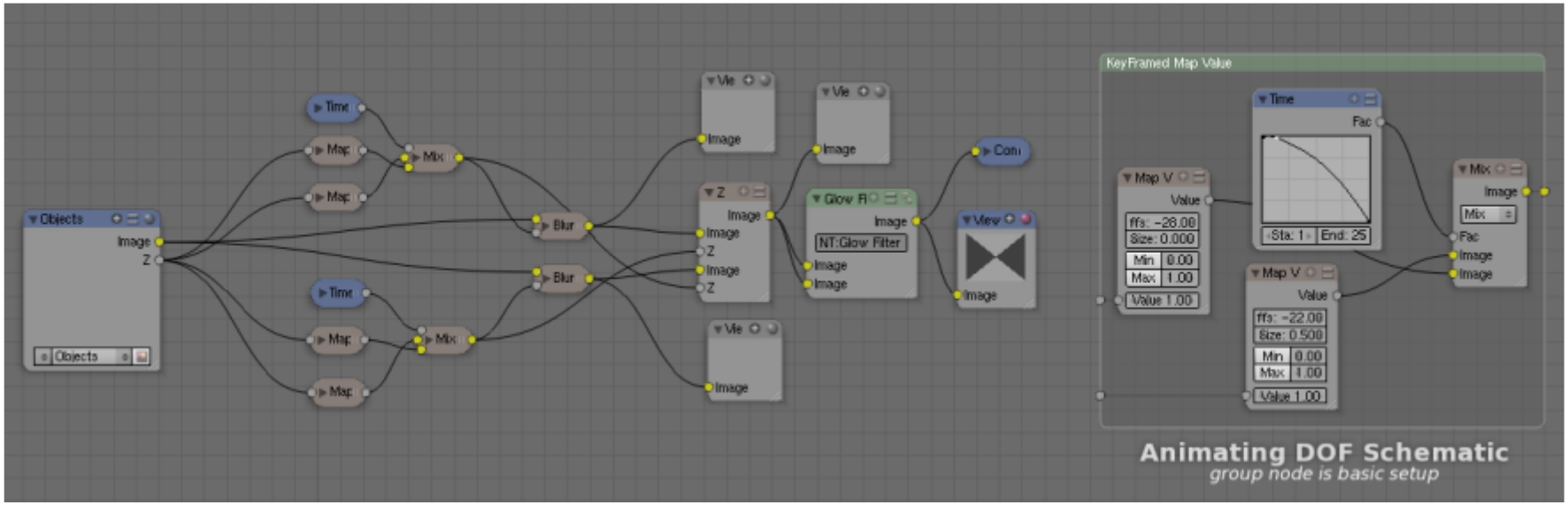


Image4 DOF Noodles for Animation

Special thanks go to Cehuken of BlenderArtists who helped explain the node setup for a similar DOF approach to the Blender community.

I'm 17 and have been a freelance since I was 15 doing everything from website design to graphic design... Nothing too important!
Artist [www.idstudios.org]
Writer [www.blendernation.com]
Developer [www.monsterweb.net]

Creating Realistic Environment For The Blender Game Engine

- by Jhon Allie

Blender's game engine is fairly simple as such things go, but with a little hard work one can still create relatively realistic games. In this article, I'm going to try to help you learn how you can use these features to make your Blender games as realistic as possible.

Planning

Before you begin making a game scene, you should have a set idea in your mind of what you want the scene to be. Decide:
What does the area look like?
What is the player meant to accomplish here?
What is the area's "real" purpose? (i.e. - if the area really existed, what purpose would it serve?)

Once you've worked out the answers to these questions, you're ready to get to work on your scene. Some people prefer to make sketches of their game scenes ahead of time, but I usually just go straight to Blender.

When designing your scene, remember that the player needs to be fenced in somehow. No matter what, people always seem to want to go beyond the confines of your game map, and you'll need to come up with a good excuse as to why they can't. Preventing the player from leaving the main part of the scene is not enough, you must also convince her that she does not want to leave it.

The most popular forms of environment bounds are as follows:

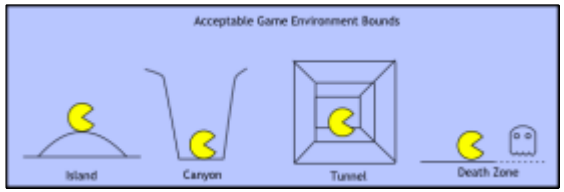


Image 1 Different environment bounds

Island is usually what it sounds like. This is an extremely popular method of defining scene boundaries: simply put in a vast ocean and people immediately assume they cannot traverse it. In my game The Voyage of the Golden Arm, I used the Island technique in a desert as well: one of the game's levels takes place on a large hill, which is surrounded on all sides by a desert wasteland. The effect is the same though the "ocean" is an ocean of sand and not water. Another variation is the "butte," in which the game area is atop a high cliff, surrounded on all sides by vertical drops. Islands can also be combined with invisible walls, to prevent the most determined players from walking too far into the ocean.

Canyon is an environment which is surrounded on all sides by insurmountable walls. Preferably, these should be high walls: short fences tend to frustrate people, as in real life they would simply step over them. The best Canyon walls should be much higher than the game character's height.

Tunnel was popularized in games like Doom, where all the action takes place inside closed-in corridors. Tunnels have fallen out of favor because they allow for very little variety in your game scenery.

Death Zone is the least recommended of the acceptable boundaries. A death zone is an area outside the main game environment which causes instant death if entered. An example is the dark caverns of the text adventure Zork, in which wayward players are devoured by "grues." These are not recommended because they seem unfair to the player, and cost lives. If you decide to use a death zone, make sure that the player is well-informed of the area's status. Nothing is more frustrating than a "surprise" death zone.

In this article, we will be working toward a realistic Island scene. I'm not expecting you to follow along with the tutorial, simply read it and look at the steps I take to improve the island's realism. Learn these techniques, and you'll have all the tools you need to work on your own creations.

Here's the little island we'll be improving:

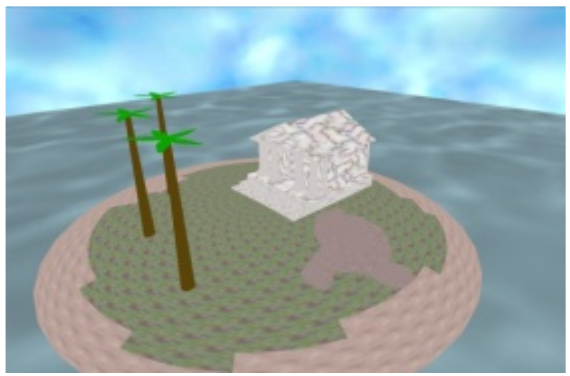


Image2. The Island scene

In all likelihood, your game art is not as bad as this, but this does demonstrate quite a few of the problems I've seen in beginner art. We'll use techniques of shading, texturing, and modeling to make this island more realistic.

Shading

Shading is how we simulate lighting in the game engine. Shading defines light and darkness on your models, and is one of the most important tools for adding realism.

The easiest way to shade your models is to make them light-sensitive. Select the model you want to make light-sensitive, press F to enter UV face-select mode, and press W to open the Specials menu. (It's important to memorize hotkeys, especially unintuitive ones like this one.) Select Light from the Specials menu.

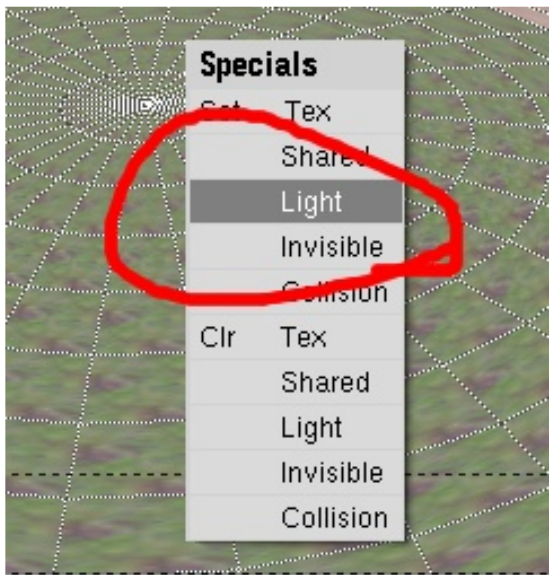


Image3 Enabling Light Sensitivity

Our horrible little island is more realistic already! Real islands aren't faceted like this, though.

Now that it's smooth, the island looks much more

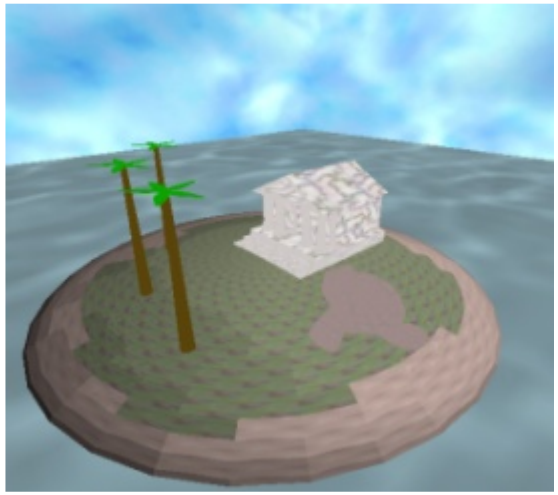


Image4 Light-sensitive island)

Image5 Enabling Smooth Rendering

real. Here's what it would look like if we set Light

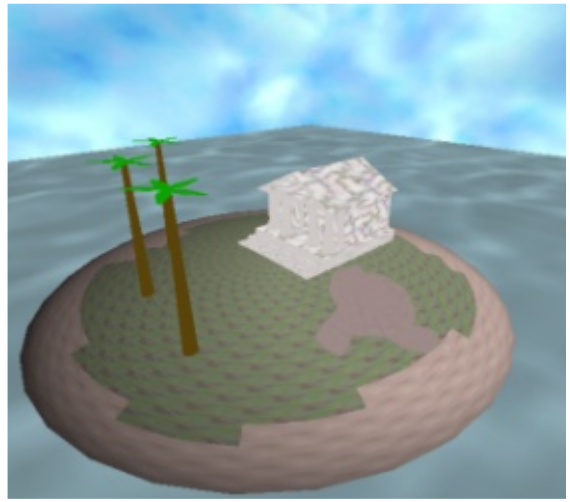


Image6 Smoothed, light-sensitive island

for all the objects:

Light sensitivity is not a good solution for inanimate objects, though, because it eats a lot of processor power.

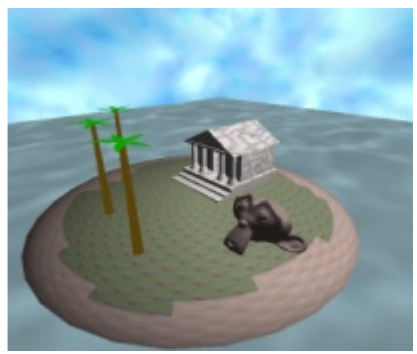


Image7 Real-time shading

Blender has another light-simulating feature, though, which simulates the light only once, then bakes it to the object's vertex colors. Simply select the object you wish to apply the effect to, and click the VertCol bake button in the Edit Buttons.

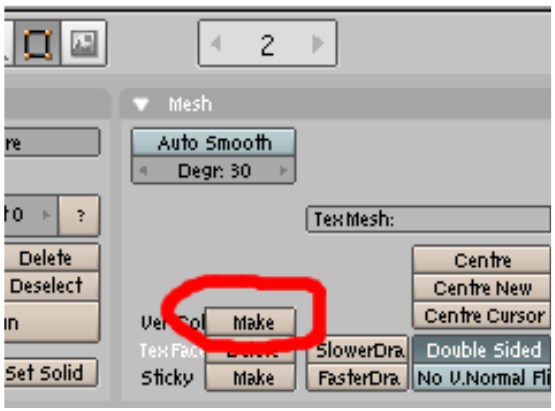


Image8 Baking Vertex Color Shading

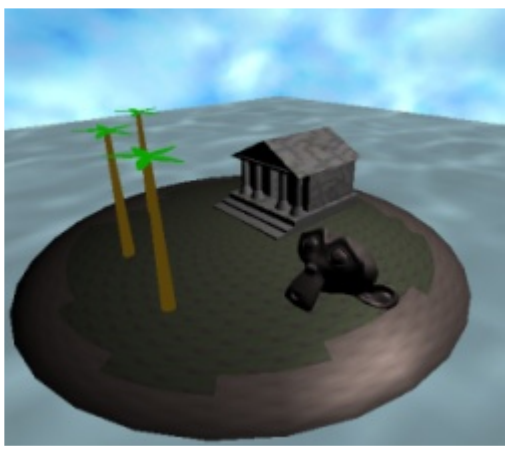


Image9 Baked Shading

The VertCol baking option is great when you're in a hurry, but the result tends to be kind of dark. Adding more lamps can help remedy this problem, as the values are calculated from the scene's lamps. Here's the same scene, with vertex colors baked from two lamps:

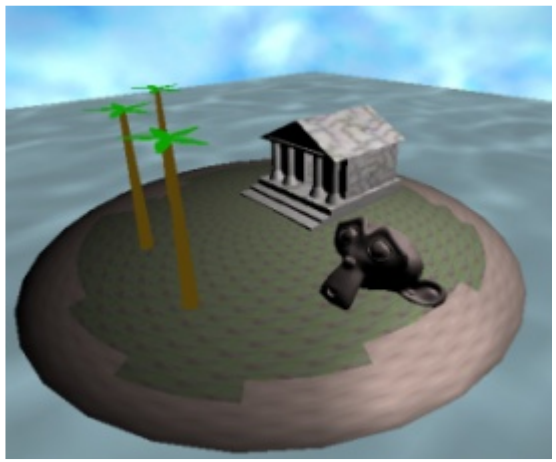


Image 10 Baked Shading with Two Lamps

The last shading method is to paint the vertex colors by hand. This is my preferred method because it gives much more control, but it does take much more effort. If you decide to shade your models this way, make sure that you try to simulate the light correctly. Try to remember where the light is coming from, and shade your models accordingly.

In general, when painting rounded models like Suzanne, I use the Vertex Paint brush. With angular models such as buildings, I select the faces individually and fill them using the Set VertCol button. Sometimes, I'll also add a slight gradient with Vertex Paint.

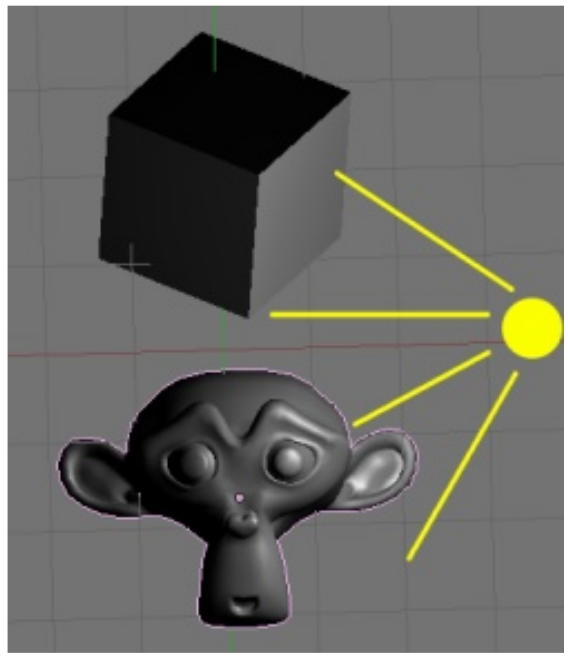


Image 11 Remember Light Directions!

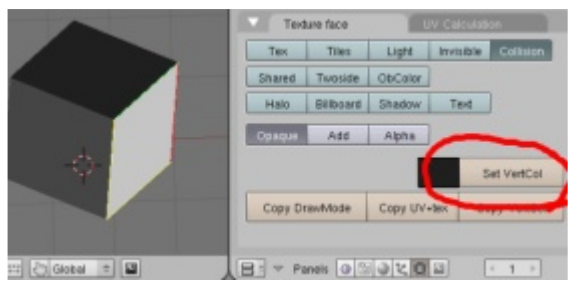


Image 12 Setting Vertex Colors By Hand

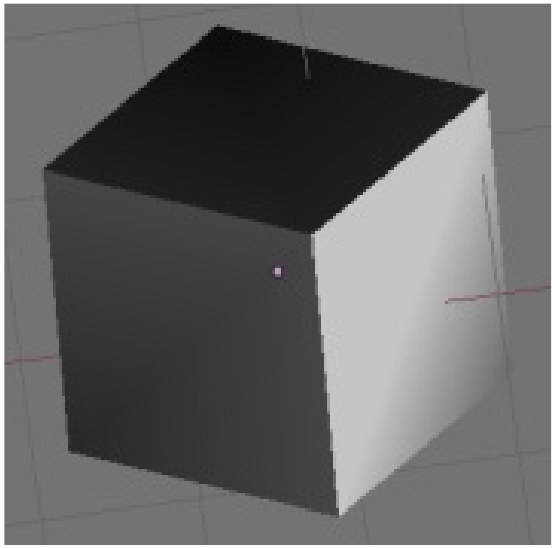


Image 13 Hand-painted gradients

To learn how to shade well, pay attention to your surroundings. Try to learn which areas are brightest or darkest. You may want to run some Radiosity or Ambient Occlusion simulations in Blender for research. (Don't use Radiosity output in the game engine, though - the models it generates are extremely complex!)

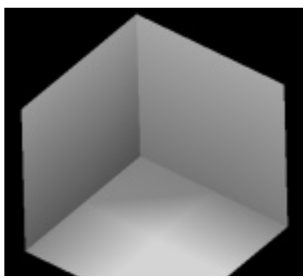


Image 14 Darkening corners can improve realism

Careful use of vertex colors can do wonders for your game environment. In the real world, one never encounters an object without shading, so it's important to keep this in mind when making your game environments. Shade everything, no matter how small it is!

Modeling

Now we'll look at how to improve your game modeling. Let's take a look at the wire mesh of our ugly little island.

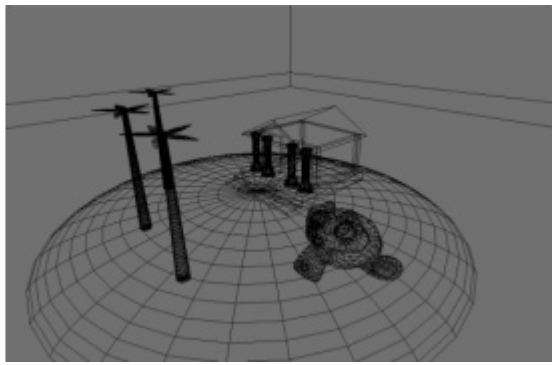


Image 15 Island Mesh

What appears to be a simple scene is not very simple from the engine's point-of-view! The island is made from a complex UV sphere, the monkey head has been "subsurfed," and the trees and building columns are both made from cylinders at the default setting, 32. It's important to remember here that though Blender can be used to make games, its primary purpose is creating rendered imagery.

This is why there are many Blender features which are incompatible with the game engine, and many more which though compatible should not be used.

Subsurf is an example of the latter. Though subsurf

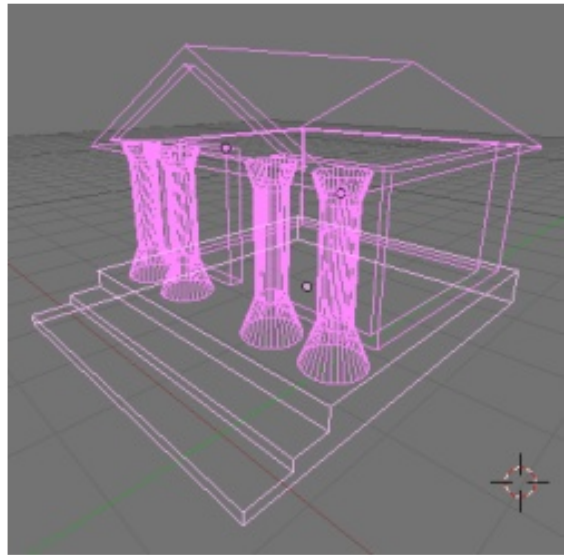


Image 16 Temple Mesh

is now compatible with the GE, it should not be used on game models. The intention of game modeling is to create "low-poly" models which are as simple as possible. Subsurf works by turning every face of your model into many smaller faces, and is therefore an extremely counterproductive method when it comes to the game engine.

When working in the game engine, your intention should be to keep your models as simple as possible: avoid using anything which makes your models more complicated.

Let's get started on our game models by improving the "temple."

We'll fix the columns first. The columns, as you can see in this image, appear to be almost a solid color. When your game models look like this, it's a fair bet you've used too many polys. Thirty-two, the default setting for cylinders, is overkill for the game engine. Ten vertices should be enough, you can probably get away with only 7, and you should never need more than 15 or 20.

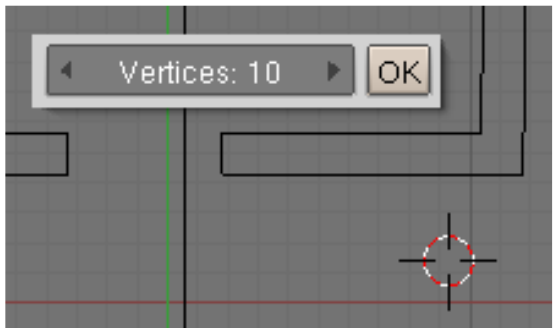


Image 17 Setting number of vertices for a cylinder

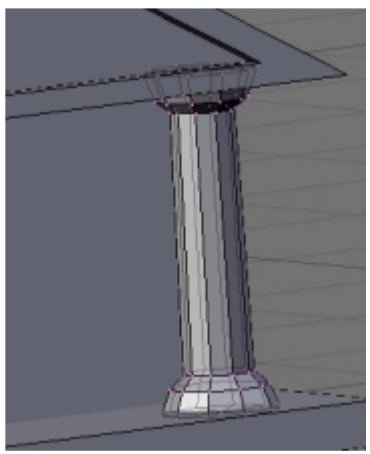


Image 18 New column from 10-vertex cylinder

We'll create a new column model and set it to 10 instead of 32.
As you can see, a column with 10 vertices still looks very round. It's not important to make everything in your game have a "perfect" shape. People expect game objects to have slightly jagged shapes, and the mind can easily fill out the model to its correct shape.

The trick is to find the happy medium between too many vertices and not enough. Ten is just about right for this column. For a bigger column, 20 might be more appropriate. For very small objects, you may want to reduce the number to four or even three. Avoid 6- or 8-sided cylinders, though: the mind is familiar with these shapes and your player may have a hard time ignoring them.

Low-poly modeling also means removing anything that is not needed, which includes faces that can't be seen. This column has faces on either end which cannot be seen. Click the "face selection"

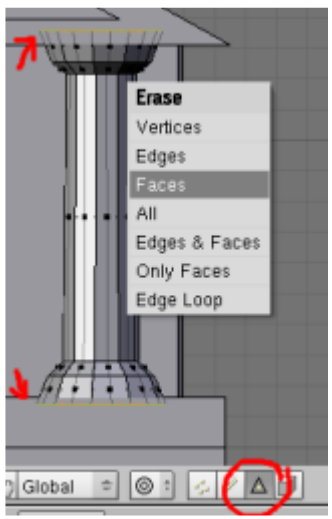


Image 19 Deleting Hidden Faces

button, select the hidden faces at both ends with the B-key select mode, press the Delete key, and click Faces.

Remember to check for these faces! They're very common, and they do tend to add up. Unimportant as they may seem, having large numbers of them can lead to a game slowdown.

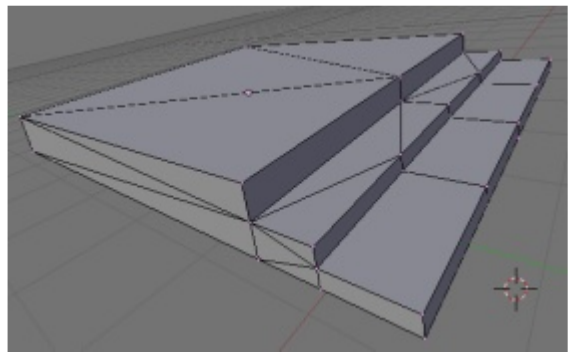


Image20 Carelessly made temple steps

Consider this:
Each column has 10 faces on each end, none of which are visible to the player. With all four of the building's columns, there are 80 extra faces! Always remember to check for and eliminate game engine waste.

Now, let's take a look at the temple steps.

Sometimes, we need to divide faces into smaller faces in order to create something. One way to create more faces is to use the Subdivide tool, but this tool divides every face into lots of smaller faces, creating a lot of unnecessary extra faces and adding to game engine waste. A much better option is to use the Knife tool.

The Knife tool's purpose is to add extra vertices to an edge. This will allow you to divide up your model's faces only as much as they need to be divided, and prevent game engine waste. To use the Knife tool, select the edges you wish to operate on and press Shift+K. The Knife menu will appear. Select Exact Line for now, and then draw a line through your edges and press Enter.

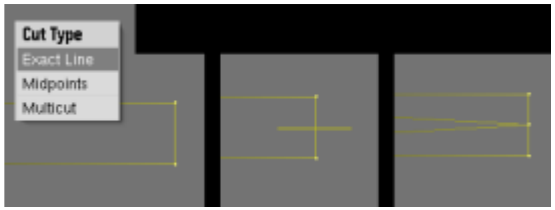


Image21 Using the Knife tool

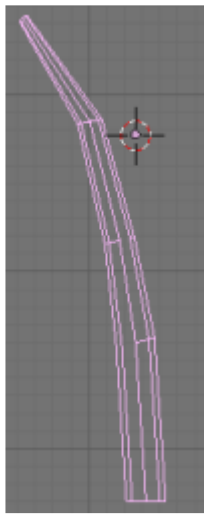
Our face is now cut in half, and ready to be extruded into steps! If we had used Subdivide, the face would have been divided into four sections, far more than we actually need. Now we can extrude the face and finish creating our staircase.

Image22 Palm Tree

The result will be a base for the temple with fewer than half the faces of the original. Don't forget to delete the underside of the temple's base: it will be underground, so there is no reason to save it.

The temple is pretty good now! Let's leave it alone and take a look at those obnoxious palm trees.

This is not a palm tree. This is a cartoon of a palm tree, and a very high-poly one at that. The trunk, like the columns on the building, is a 32-vertex cylinder. It is absurdly straight: the natural world simply does not have such straight lines. Our first order of business will be to improve this trunk.



Add a 7-vertex tube and resize it until it's long, straight, and narrow. Scaling it small then enlarging it on the Z axis should work. (Make a habit of using tubes in place of cylinders when neither end will be visible. In this case, one end of the trunk will be below ground, and the other will be above players' heads. Neither will be seen, so you can use a tube in place of a cylinder and save yourself some polys.) Once your tube is ready, use the Knife tool to slice it up a few times, then give it a subtle bend. Scale it down a bit toward the upper end.

Image23 New tree mesh

This trunk is not only more ealistic, it has fewer polys as well!

To create the leaves, add a plane and extrude once from each end:

Now, select your ends one at a time, press W, and select Merge.

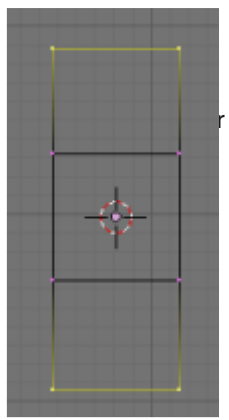
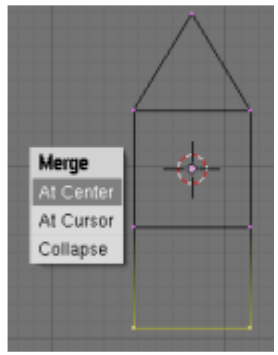


Image24 Creating Palm Leaves, Step 1



The result will be a plane with a triangular face attached at each end. Now use the Knife tool to divide the plane vertically down the middle.

Select the new vertices and move them up so that the leaf will curve in the middle.

Image25 Using the Merge tool, Step 2

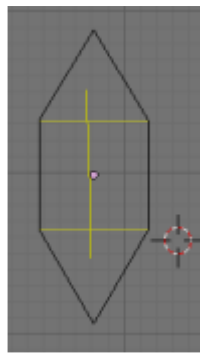


Image26 Creating Palm Leaves, Step 3

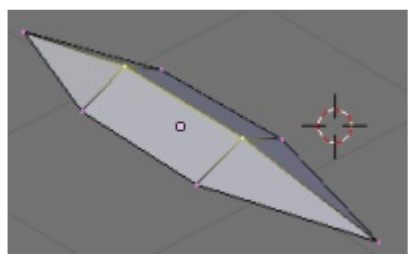


Image27 Completed Palm Leaf mesh

This is our basic leaf model. You can tweak it a little if you like. It may not look very much like a palm leaf yet, but the texture will improve that. It should be noted that since leaves are close to being flat, it is perfectly acceptable, and recommended, to portray them with plane-based objects like this one. In the game engine, try to portray complexity through texturing rather than modeling, whenever possible. This palm leaf does not need to be very complex, because its true realism will be applied through its texture. We will be covering texturing shortly, but I'll jump ahead just a moment to show you what the completed palm tree looks like.

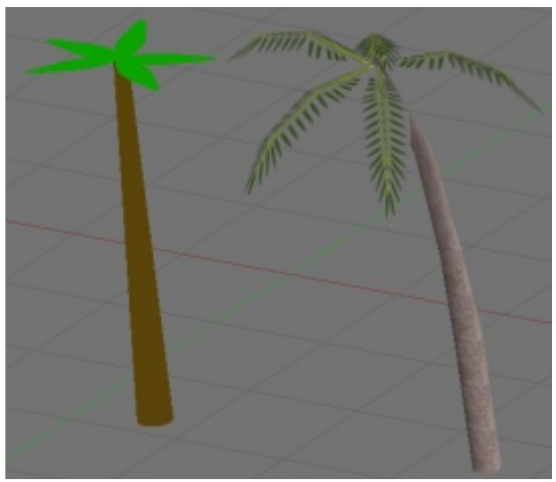


Image28 Original & Improved Palm Trees

Look how grotesque the original palm tree looks, now that it's shown beside one that's so realistic! It's always worth going the extra few steps to make your game look good. Don't be afraid to invest the extra time - in all likelihood, your game will not only look better, but it may run faster as well.

The large Suzanne head I will simply delete. Always try to make your own models: they'll more closely resemble your own style, and they may even be more compatible with the game engine. There is no need for a giant stone monkey head here, though, so there is no reason to include it. If you are making a Blender-themed game, however, incorporating Suzanne is encouraged!

Let's spend some time rebuilding the island itself. At the moment it's just a big round thing. That's fine if your game is about The Far Side, but if you're aiming for something more realistic, we have to improve it a bit.

This is one time when I consider the use of Subdivide to be justified. Add a plane, scale it up a bit, and subdivide it a few times.

The illustration shows the plane after four subdivisions. That should be enough. Now press [O Key] or open the menu to activate Proportional Falloff editing mode.

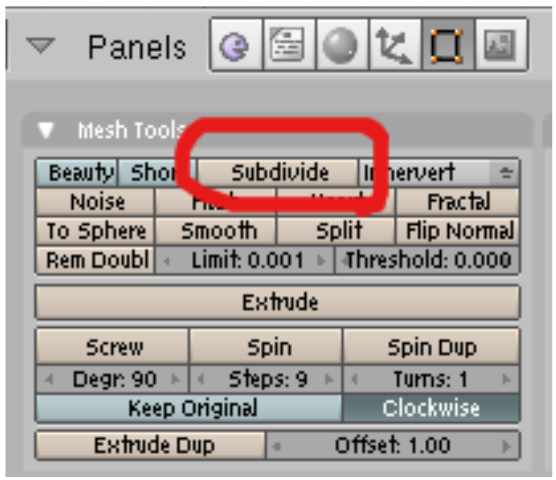


Image29 Subdivide button

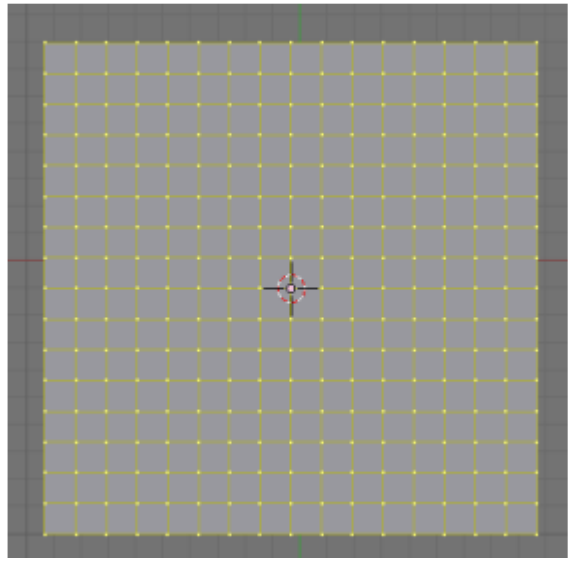


Image30 Subdivided plane

Image31 Proportional Falloff Menu

Now select a few vertices and begin moving them. You will see that nearby vertices will begin following them along. You can move the mousewheel to adjust the extent of the effect. Pull up the ground to make some subtle hills. I pressed Z while translating to lock movement to the vertical axis.

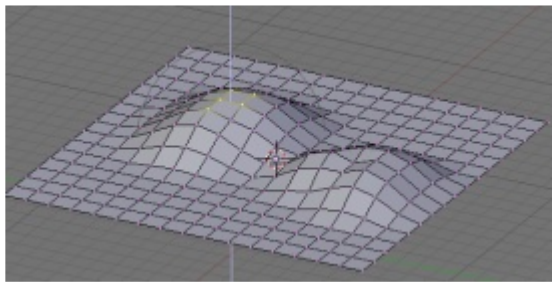


Image32 Using Proportional Falloff to Make Hills

Your completed island should look something like this:
Note that the highest points of the island are in the

Image33 New Island Mesh

middle, and the edges slope downward into the ocean. This is actually a pretty good island model, although the steep slopes might make it difficult to navigate. It's also worth noting that some video cards (many Intel chipsets, especially) experience a strange problem which causes landscapes like these to "squirm." If you experience this problem, simply select the entire model in edit mode and press Control+T to triangulate the mesh.

One last thing to look at before we move on to texturing - the "Skybox."

As you can see, the cloud texture is literally

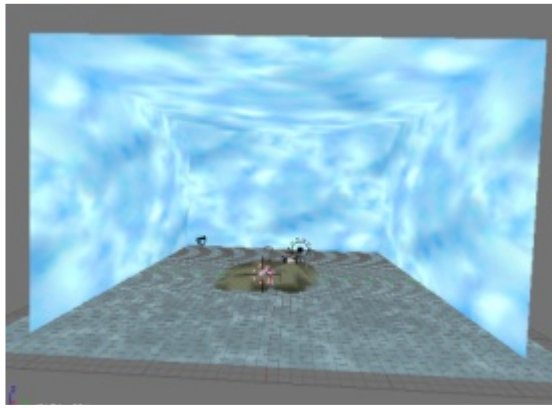


Image34 Ugly Skybox

mapped onto a giant box. Some people can make this technique work very well, but I've found that it's extremely difficult to disguise the fact that it's a huge box. I prefer to use tubes, spheres, or planes to create clouds. The rounded edge of the tube or sphere is more subtle than a cube's right angles, and therefore harder to spot and easier to hide.

The tube is a great way to make clouds. It requires a bit of extra work on your part to make its special texture, but the result is worth the trouble. You simply need to create a good cloud texture, and then use your painting program to add a color-to-transparency gradient sky color at the top, to achieve a result like this:

Then map the texture onto the tube, and set Blender's world color to the same sky color as the image. The result: Pretty nifty, eh? Not only does the Tube method create nice,

Image35 Cloud Texture for Tube

seamless clouds around the horizon, but it hides the corners of the ground or ocean plane as well. A nice way to animate Tube clouds is to make the

Image36 Tube Clouds

tube rotate very slowly. The effect is subtle but effective.

The Sphere sky method works basically the same as Tube, but I find that it's harder to map textures onto a sphere, and they have more polys as well. The advantage of Sphere is that it allows clouds to be on all sides of the scene.

The Plane method is fairly simple. You can simply add a plane above the scene and cloud-texture it. When I use this method, I often create an object like the one on the left, and vertex paint it as seen on the right:

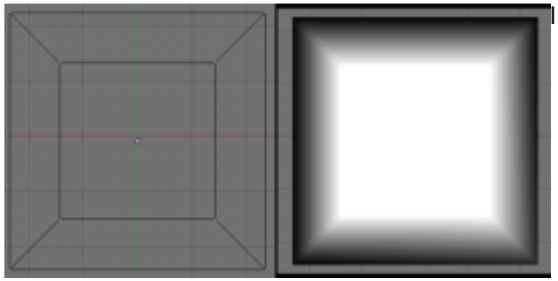


Image37 Planar Cloud Object mesh and vertex colors

then map it with a black-and-white cloud texture, and set it to Add in Face-Select mode.
These settings create simple clouds which fade

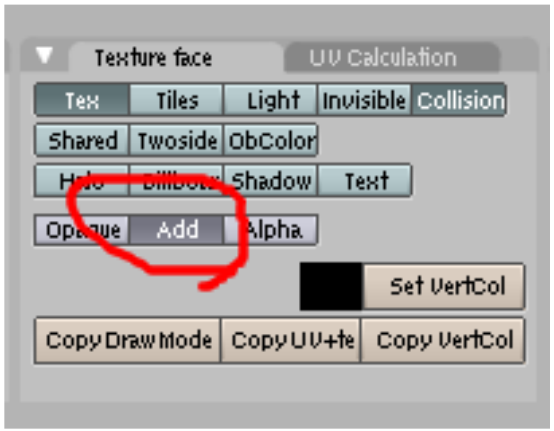


Image38 Enabling Add Rendering

out around the edges. They have the appearance of all Add-rendered objects, but you can always change that by creating a texture designed to be opaque instead.

Note that plane-based clouds appear only above,

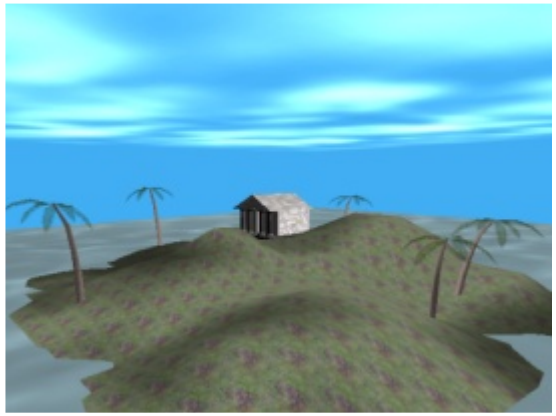


Image39 Planar Clouds

whereas Tube clouds appear only at the horizon. Sphere can be considered "the best of both worlds," but it's a bit harder to work with. However, some perseverance can get you a nice effect.

One last word about clouds: you'll probably

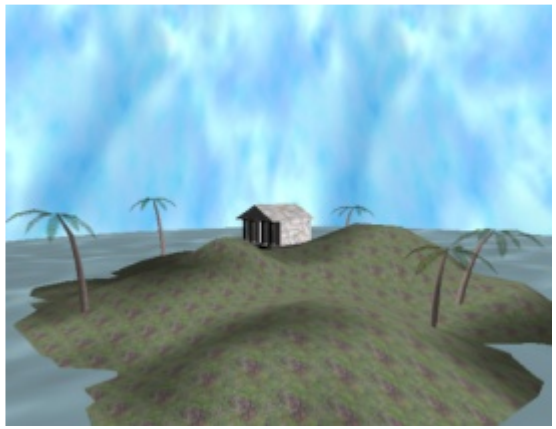


Image40 Sphere Clouds

experience some problems due to the fact that faces are, by default, visible on one side but not the other. You'll have to recalculate the cloud object's normals to make it visible from the inside. In edit mode, simply select all its vertices and press Ctrl+Shift+N. This should fix the problem.

Final thoughts about game modeling
Try to use the smallest number of polys possible. Small objects can often get away with very, very few. A handle on a door, for example, might be made out of a three-sided tube! Large objects often need a higher-resolution mesh, though: don't be afraid to give them what they need.
Resist the temptation to create high-poly models and then decimate them. Decimation creates ugly, complicated meshes which are hard to texture and extremely difficult to animate.

The Decimated version of the High-Poly Suzanne is not equal to the original. Everything will be easier if you start out modeling low-poly in the first

Image41 Why Decimation is Not Your Friend

place. Don't try to cheat by modeling high-poly and decimating it later.

Texturing

Though shading and modeling are both crucial to creating realistic game environments, your most powerful weapon by far is texturing. It does not matter how perfect your models are or how well they're shaded: if they have bad textures, they will look terrible! Texturing is the final word, it alone "makes or breaks" your environment.

I make nearly all of my textures myself. I take photographs of interesting surfaces and convert them into textures using Gimp. There is no alternative to photographic textures when one wishes to achieve realism.

The most important thing to remember when creating textures is to make them seamless. You can do this quite easily in Gimp by navigating to Filters->Map->Make Seamless in the menus. This effect is not perfect, but it does hide the seams in your textures quite nicely. You may still need to dodge/burn by hand a little to ensure a uniform shade, but for the most part, making textures is painless.

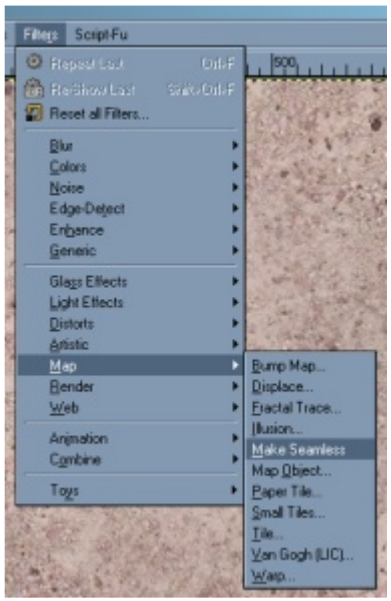


Image42 Making a texture Seamless in Gimp

I'm not going to give a whole texture-creation tutorial here. Suffice it to

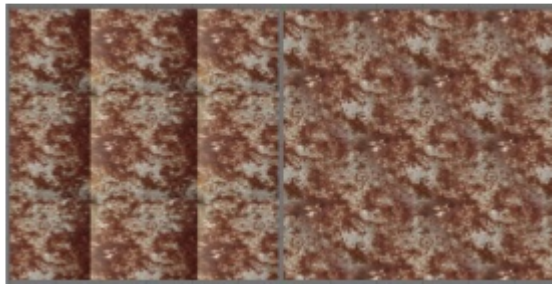


Image43 Non-seamless and seamless textures

say, texture-making consists mostly of altering photographs until they can be repeated without obvious repetition.

If you look at the island, you'll see that its texture is obviously repeating. The ground texture contains a strongly defined "purple" spot, which is repeated again and again for a crazy "polka-dot" effect.

One quick way to remedy this is to switch from the default 1/1 UV texturing setting to a different

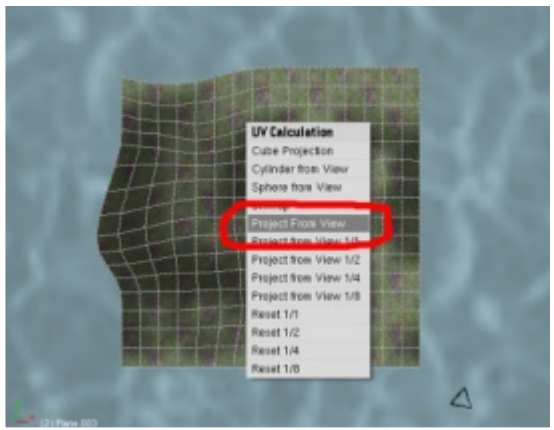


Image44 Selecting a UV mapping option

setting. In the UV face select mode, press A to select all the faces, and press U to choose a mapping option. Blender 2.42a delivers a horde of new UV mapping options, and I'll admit that I'm not familiar with all of them yet! I've found, however, that for relatively flat objects like our island, the "project from view" option works nicely. Select it while viewing the island from above, then simply scale the entire UV map up and down in the UV window until you like the result.

As you can see, the island's appearance is already

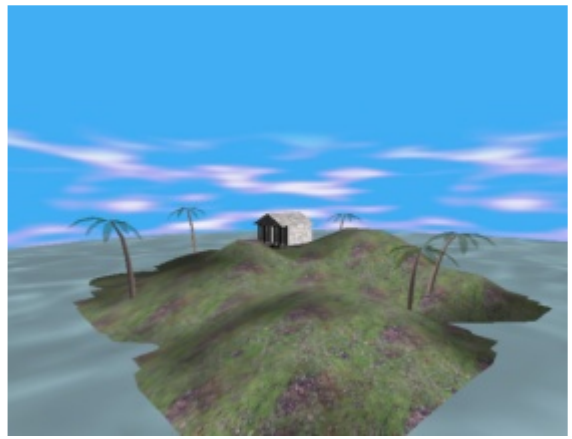


Image45 Improved UV mapping

drastically improved. The purple spots, though not gone, are not such a big problem any more. You may still want to try to change their tone, though, which can be easily achieved through some small Gimp tweaks.
The easiest, and often most effective, method is to simply select a greenish color, then brush over the purple areas with the brush tool set in "Color" mode.

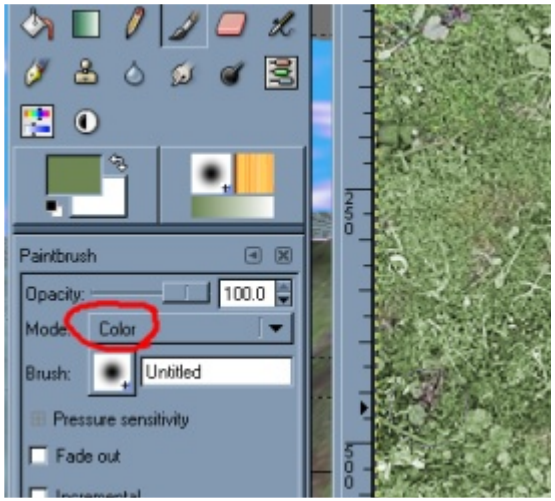


Image46 Eliminating Color Variation in Textures

Image47 Island with Improved Textures

The island looks a lot better now that the ugly purple spots have been eliminated... but before it can look realistic, we need a few more different textures on there! Let's start by adding a beach which runs around the edge.

First create a sand texture, and then create a second texture which blends the sand texture into the grass texture. It should look something like this: I'm going to assume here that you have sufficient knowledge of Gimp (or Photoshop) to do this. Though a step-by-step guide to creating textures

Image48 Grass -> Sand Texture

would be useful, it is beyond the scope of this article.

Now, map your plain sand texture (without grass) onto the faces around the edge of the island (See Image 49).

Next, is the time-consuming part. Map the "blended" texture we made to the faces in between the grass faces and sand faces. I find that the 1/1 setting works best for this. Getting this perfect can take a while, so be prepared to spend a good deal of time on it.

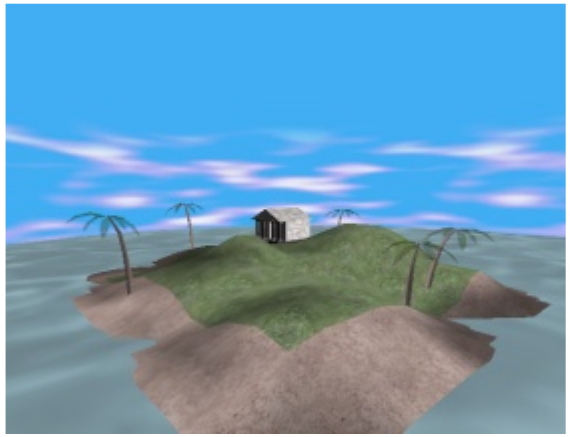


Image49 Island with Two Textures

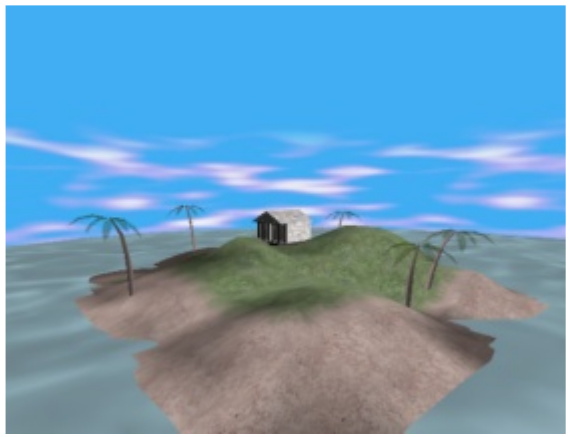


Image50 Two Textures Smoothly Blended

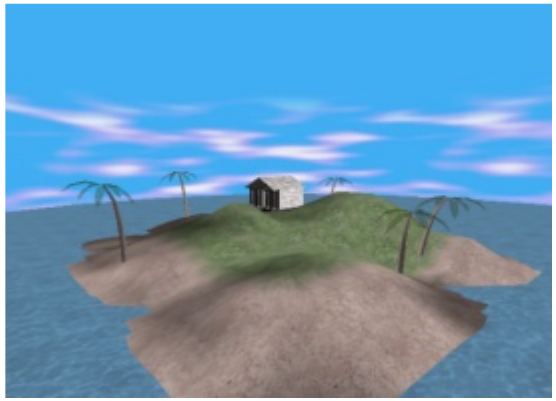


Image51 Improved Ocean

Look at how realistic the island is now compared to the way it looked at the end of the Modeling section! This shows how important textures are to your scenery. We could call this finished now, but let's apply a few more changes first.

First, let's fix the texture on the ocean plane. There's nothing wrong with the texture, but we need to scale up the UV mapping slightly to increase the detail on the water surface.

Notice how even slight changes can improve the overall appearance drastically. I also tweaked the sea's vertex colors slightly to more closely match the sky.

Now let's add some subtle waves that lap at the edges of the island. Add a plane at the edge of the island and begin extruding it so it follows the edge closely. Keep doing this until you've circumnavigated the island, then merge the ends together to create a continuous ring.

Make a nice black-and white "ripples" texture, and map it 1/1 to the faces. Set all the faces to "Add" render mode, and paint the outer vertices black.

Image52 Making Waves

These small ripples integrate the island with the sea, finally pulling together every aspect of our game scene. If you'd like, you can even animate t

Image53 Making Waves

hem with the "UVscroll" Python script.

For one final touch, you may wish to create shadows for your palm trees. A nice tutorial about creating shadow textures for games was created by master game maker ST150, and is available at

http://www.blending-online.co.uk/8501/50779.html

Closing

Maybe you didn't want to create a simple island like this one, and maybe you didn't even try to follow along. It doesn't matter: my hope is that

Image54 Completed Island

while reading this article, you gained a better understanding of how to think like a game developer. Whether or not you now have a little island scene does not matter, what matters is that now you better understand the concepts and application of game scene creation in Blender.

Plan ahead to create the best scene possible. Try to start with some idea of what you want the finished project to look like. Create concept sketches if you prefer.
Shade your models, either through vertex colors or by using light simulation. Keep your eyes open in the real world, and try to learn how light behaves.

Think while you model: never use 10 faces if three will do, and use face-dividing tools sparingly. Delete unneeded faces, and never make your models more complicated than necessary. Try to see how small a face count you can get.

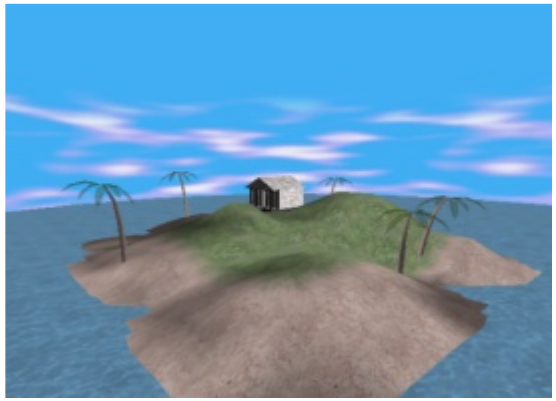


Image51 Improved Ocean

Create your own textures whenever possible. Bring your camera with you on trips, and photograph as many different surfaces as possible. When processing your textures, strive to obscure the repetition.
Never be afraid to add realistic details, especially ones which have little effect on polycount.
As a final touch, you may want to experiment with adding motion. A slight swaying to the trees or a spinning weather-vane can add to the game's look beautifully. The world is full of motion, and a little motion can be a great addition to your game. Try to make sure that the player is not the only thing moving in your world.

And that's that! I hope you enjoyed reading the article. I've strived to include in this article everything a beginning game artist might find useful. Admittedly, there are a lot of things I didn't mention, but there is enough material here to get someone on their feet and solve some common problems. With these skills in place, you'll be well on your way to creating a great Blender game. Go start up Blender, and get to work! I look forward to seeing what you create!

John Allie / PlantPerson

I have been using Blender since April 2000. My current game project is an adventure game called "Into the Titan."

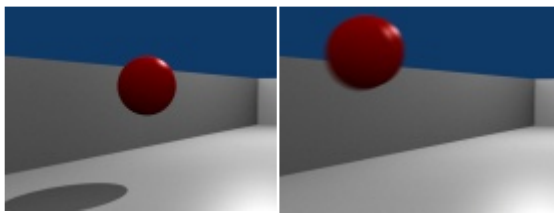
Website:
www.gorilla3d.com/plantperson

Blender And Vector Blur

-by Olivier Saraja

Introduction

Vector Blur is an interesting solution that goes beyond the limitations of regular Motion Blur. It relies on the Composite Nodes Editor to perform. But even if the Nodes Editor may look intimidating at first glance, it is in fact very easy to set up. This short tutorial will go through the basic steps for enhancing your renders (both stills and animations) with good looking motion blur.



The main thing to understand is that the object, to which the Motion Blur will be applied, should have been animated beforehand. This means that the object must follow a curve or be animated using IPOs. In fact, the rendered picture will be turned into a 3D model with depth, and a speed vector will be calculated for each pixel of the picture. As a post-process, the Composite Editor will add the motions to the final pic in a blurred way. This is why defining the motion of the object is absolutely mandatory.

A very brief example file of such an animation can be found at the following URL:
http://feeblemind.tuxfamily.org/dotclear/images/atelier-improbable/vector

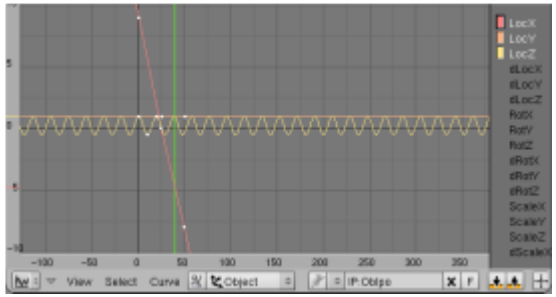


Image 1 An example of IPO curves affecting a moving object.

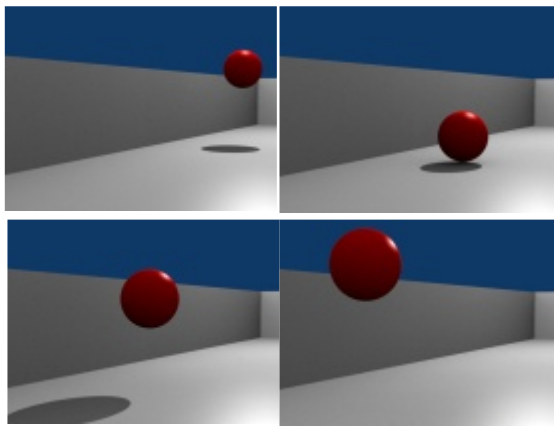
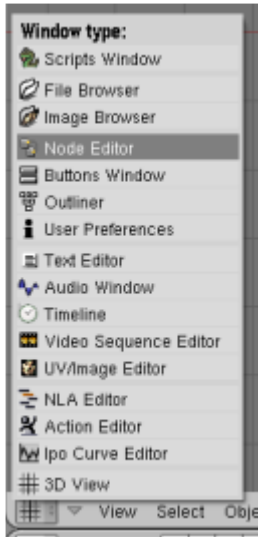


Image2 Here's how it would look on a regular animation rendering, without vector-based motion blur

A step-by-step of the Vector Blur process

First, you will need to turn one of your 3D views into a Node Editor view. This is done by clicking on the grid-like icon at the left end of the Header of the view. A menu will pop up and let you define the type of view needed. Choose the Node Editor.



In the Header of the Node Editor, you will find a menu for the view (View, Select, Add, Node) and then two icons: (the grey sphere icon) enables Material Nodes, and (the small face picture icon) enables Composite Nodes. Click the Composite Nodes.

.

The default Composite nodes network should be displayed on the screen. Using the [LMB], draw a box that will cross the link between the Render Layer node and the Composite node: the link should disappear. Now, use the Add menu: Add>> Filters

Image3 Turn one of the views into the Node Editor

Image4 Click on the Composite mode icon and then activate the Use Nodes button.

>>Vector Blur. A new node should appear, titled Vector Blur. Move it between the two previous default nodes, and start the magic: connect the output Image, Z and Speed from the Render Layer node to the input Image, Z and Speed from the Vector Blur node, using the [LMB]. Once done, connect the output Image from the Vector Blur node to the Image input of the Composite node.

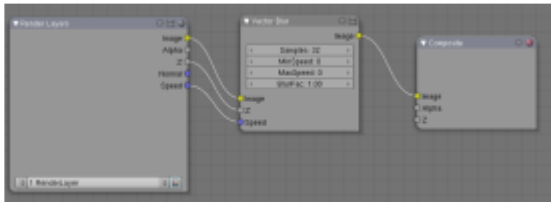


Image4 The Composite nodes are set to deliver a Vector Blur based rendering

But, like anything else in Blender, taking into account the effect of the Vector Blur is nothing but an option. You need to tell Blender that it must do the post-processing simulating the motion blur explicitly. This is done in the Scene [F10] menu, in the Render buttons, in the Anim panel. You will have to activate the Do Composite button. As a final pre-requisite for Vector Blur motions, you will need to activate the Vec option in the Render Layers tab.

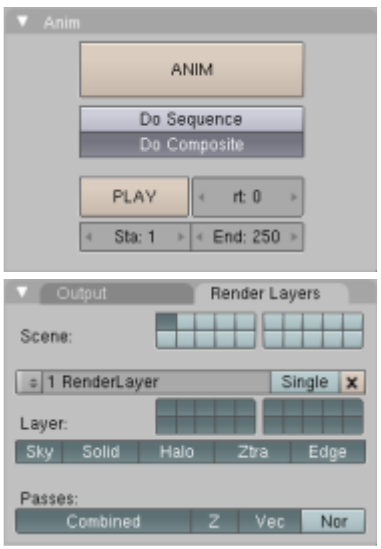


Image5 Do not forget to activate the Do Composite button in the Anim panel and the Vec button in the Render Layers tab

You can now render your picture in two ways:

- You obviously already know the [F12] shortcut or the RENDER button in the Render panel
- You can also render the picture using the tiny Render icon in the Render Layer node

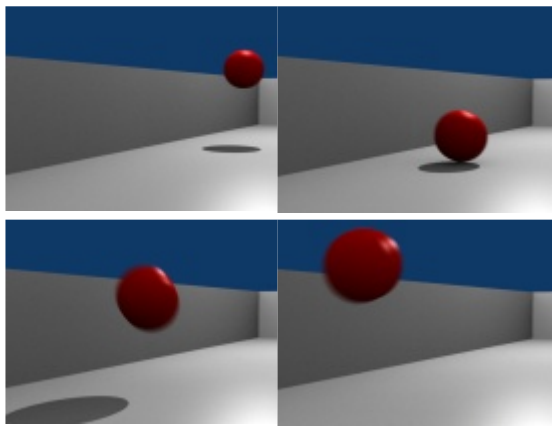


Image2 Here's how would look the same animation rendering, with vector based motion blur enabled

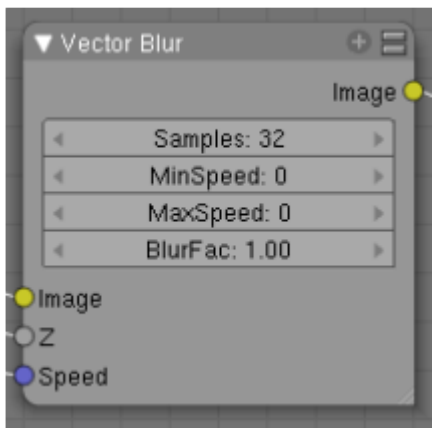
Some explanations on the Vector Blur node
This node only has a few settings; most of them could work quite well with their default values. But sometimes (if not every time!), you need to go beyond default settings to get real cool results. There are four settings to know about.

Samples - This setting controls the blur effect intensity: the higher the samples, the more blurred the object will look.
MinSpeed - If everything is blurred, but some objects are a lot slower than others (or totally static, as the background of the pic, for example), then this parameter will help differentiate high-velocity objects from null-velocity (or slow-velocity) objects. This is really useful during camera movements or slight background movements.
MaxSpeed - If you have extremely fast objects in the scene but the blur does not render well enough, then you can make use of this parameter to better the vector blur. Note that a 0.0 value means that no maximum is used.
BlurFac - This parameter will scale the vector speeds, calculating from the movements of the object. The visual effect is close to the shutter speed of a regular camera.

Concerning the use of Vector Blur, a good tip is to put fast moving objects on a Render Layer, slow or static objects on another, and to specify independent parameters for each, for better control.

Moreover, large objects not entirely seen from the camera should be subdivided so that at least some normals of the object are actually seen from the camera, or the blur effect could be drawn on the wrong side, compared to the movement.

There are some limitations that you should know about.



Since Vector Blur is a post-process effect:

- if an object is vector blurred, it's shadow is not blurred!
- if a vector blurred object is moving behind a (ZTransp or RayTransp) glass, it won't appear blurred through the glass
- if a vector blurred object is moving in front of a (EnvMap or RayMir) mirror, it won't appear blurred in the reflection

Olivier Saraja

Olivier Saraja is an Open Source enthusiast running on Linux since 1999 or so. He uses Blender since version 1.63a and always dedicated himself to writing tutorials for the community, both in french and in english. He has just written a french book about Blender: "La 3D Libre avec Blender" (which could be translated as "Free 3D with Blender") at Eyrolles Editions:

http://www.editionseyrolles.com/Livre/9782212119596/la-3d-libre-avec-blender

He is also a casual Wiki contributor of Blender's documentation.

Blender & Displacement Mapping for Beginners

-by Michael Wach

Introduction

Bump mapping and Displacement mapping are two special techniques for making an object appear to have a rough or irregular surface.

What is Bump mapping?

Bump mapping takes a grayscale image and reads the light and dark information to simulate an irregular surface. When you render an object with a bump-mapped material, lighter (whiter) areas of the map appear to be raised and darker (blacker) areas appear to be lowered. Note that bump mapping does not modify the geometry, only the normals.

The bumps are a simulation created by perturbing face normals before the object is rendered. Therefore, bumps don't appear on the silhouette of bump-mapped objects. Bump mapping is useful for adding detail to an object without increasing the polygon count.

What is Displacement mapping?

Displacement mapping is very similar to bump mapping; where a two-dimensional image's grayscale information is used to change the appearance of a three-dimensional object. Lighter tones create raised bumps and darker tones create lowered indentations as usual. Unlike bump mapping, which only affects the object's texture, displacement mapping affects the object's geometry, creating additional polygons according the displacement map's properties.

Displacement mapping can be very demanding on your computer's performance as it tends to create an extremely large amount of polygons. This high resolution object can appear much more photo-realistic than bump mapping, if used properly. As computers are becoming more and more powerful, displacement mapping may soon be able to render at speeds necessary for a reasonable game engine. However, current technology does not meet the heavy processing requirement.

Before you start

Bump mapping & Displacement mapping techniques are very fast and easy to do in Blender, even if you are new to Blender's texturing tools. Please note that displacement maps are not displayed in the 3D view port. Bump maps can display a very basic preview with shading [Shift+Z] turned on.

Bump mapping in Blender

For this tutorial, I will use one of Blender's many procedural textures, "Musgrave". Feel free to pick another type, or even use your own bump map image! Begin by opening Blender and creating a new object. I used the popular "Suzanne" (monkey) model for this example.

Next, create a new material in the Material Shading Panel [F5], and add a new texture. Now switch to the Texturing Panel [F6], where we will create the bump map. From the "Texture Type" box, select "Musgrave". I left the default settings alone, but you may choose to play around with them. You can also add a ramp-shader here if you are planning on adding unique colors to the texture.

Image 1 Regular Texture

Image2 Bump Map

Image3 Displacement Map

Now, we need to configure the most important settings for the bump map. These controls are located in the Material Shading Panel [F5] under the "Map To" tab. You may choose to disable the texture's color (Col) when modifying your bump or displacement maps. This can sometimes give you a better idea of what's going on with each setting.

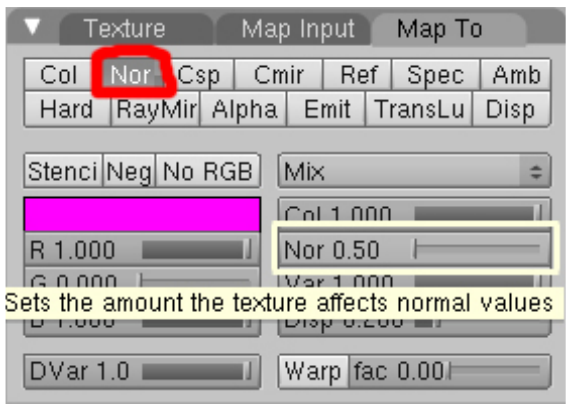


Image4 Enabling Nor Mapping

To enable bump mapping, press the normal (Nor) button. You can adjust the strength of bumpiness with the slider button labeled "Nor".

Use bump maps when you want to take the smoothness off a surface, or to create an embossed look. Keep in mind, however, that the depth effect of a bump map is limited to the normal (Nor) slider.

Displacement mapping in Blender

The way you create a displacement map is very similar to the way you create a bump map. The only difference is, you must enable the displacement (Disp) button. You may turn off the normal (Nor) button depending on your desired effect.

The displacement slider controls how far away from the original mesh the vertices will be moved. The normal (Nor) slider also has a great impact on your displacement results. Be careful when using displacement maps because they can be very processor demanding.

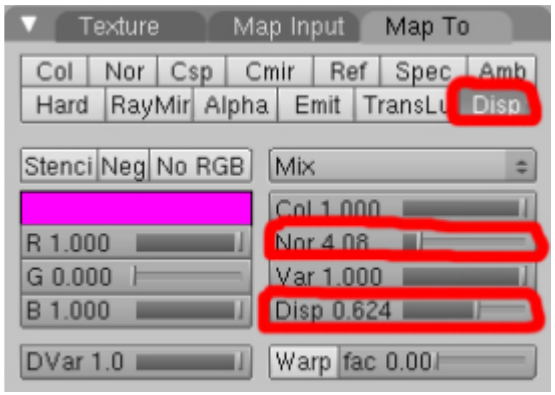


Image4 Enabling Displacement Mapping

Conclusion

Overall, bump mapping and displacement mapping are very handy tools if you know how to use them. Maps that shade between white and black generally work better than maps with hard edges between the white and black areas. I highly recommend playing around with the settings and rendering your results to help deepen your understanding of them both.

Michael Wach

Is an 18 years old from Cleveland, Ohio. Currently studying at the University of Advancing Technology. He works over internet for a NewYork City based patent attorney, creating 3D representations of his ideas using Blender 3D and YafRay. Also make websites, compose digital audio, record and edit digital videos and works with a lot of 2D illustrations, textures, and photographs.

Check www.mikewach.com or contact him at *mail@mikewach.com*

Creating A Lightning Bolt in the GIMP

-blenderart

Introduction
This time, we will learn the procedure for creating a lightning bolt sprite from the GIMP.
Lightning bolts find good use in 3D game SFX, like zapping an item or enemy from an electric weapon, denoting tension in sci-fi scenarios, or just a natural lightning bolt to enhance the environmental mood.

Level : Newbie to Intermediate

Image 1 Solid Noise pop-up

Step 1. Launch the GIMP and create a new 256x768 image. Go to Filters>>Render>>Solid Noise. In the pop-up, enter the values as shown in Img 1 or, choose one of your own by hitting 'New Seed' button. Finally, press the OK button to render the noise.

Step2.
Create a new layer over the current layer. Go to the Tool Box and select the gradient tool and in the Tool Options, select the shape of the gradient as Bi-linear and enable 'Adaptive-Supersampling'. Create a new layer over the noise layer and draw a small gradient over the new layer (see Img3)). One thing to remember here is that you should try to draw the gradient over the part of the solid noise which contains mostly white and it's surrounding areas are darker. That is where you get good contrast for the light bolt.

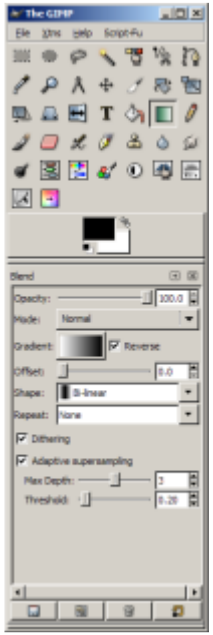


Image2 Gradient settings

Step3.
Go to the 'Layers' dialog box and change the layer-mode of the gradient layer to 'Difference' see img 4. Immediately, you can see the formation of the bolt. Now go to Image>>Flatten Image to flatten all the layers into one single layer. Now go to Layer>>Color>>Invert to invert the color of the layer.

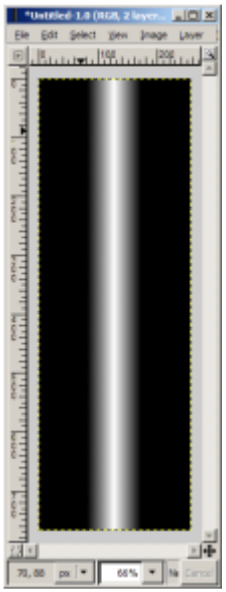


Image3 Drawn Gradient

Step4.
What you get is a streak of two lightning bolts (See img5), but these bolts are hardly usable as they appear virtually glued together. Go to Layer>>Color>> Levels. This will bring up the Level adjust dialog box (see img6). Drag the input levels mid-pointer towards the right.

While you do this, you can also see the formation of two neat lightning bolts (See img7). Adjust the amount of sharpness you want then, press 'OK'.

Image4 Layers Dialog

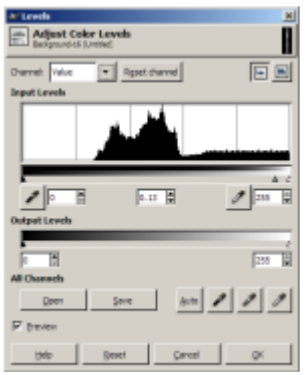


Image6 Level adjustment

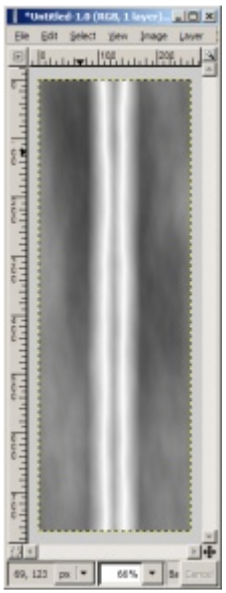


Image5 Lightning bolts

Step5.
To separate the lightning bolts as an Alpha channel, you will need to convert the black color to alpha. To do that, go to Layers>> Transparency>> Color to Alpha.
In the Color to Alpha dialog, click the 'From' button and choose a black color then press 'OK'. You now have an alpha-based version of the lightning bolt (See img8). The last thing to do is save it in a file format that supports transparency, like PNG or TGA.

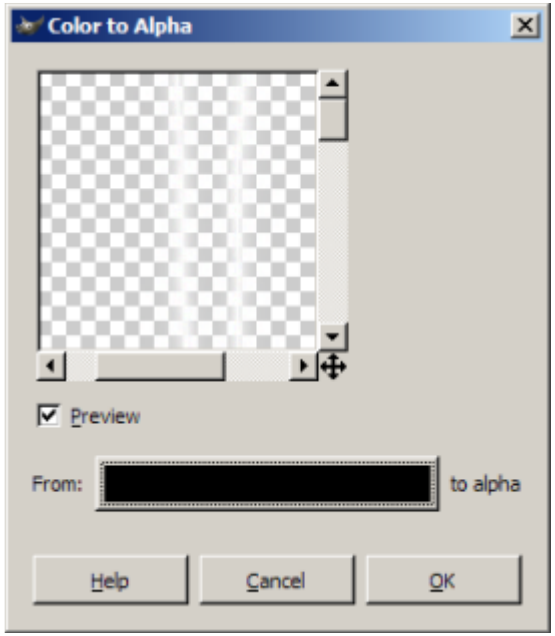


Image8 Lightning bolts with alpha channel

Conclusion
Although this method only generates pretty straight lightning bolts, you can compliment it with the noise generation plug-ins for the GIMP called 'Plasma2' etc. This will allow you to have random noise generation which can result in more varied lightning bolts. I hope this has provided you with a nice and easy way to create lightning bolts.

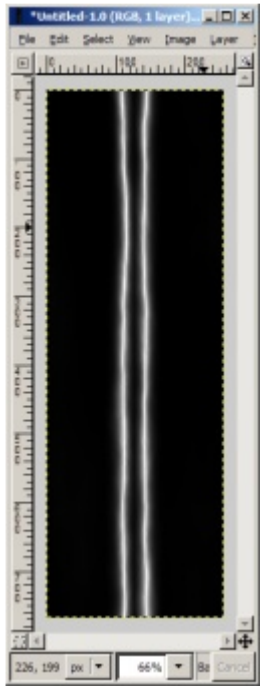


Image7 Rendered lightning bolts

A Compositor Node Overview

-by Daniel LaBarge

Sandra Gilbert and I discussed the fact that with all of the new features coming from the Blender releases, it is often hard to keep up with them. Often, you never actually get to use the feature in a project because you are simply too busy exploring all of the different ways of using the new functionality. To help those get up to speed faster, I'll give a quick overview of the Compositor Node system.

NOTE: The description of how each node works is based only on the utility and not on the actual mathematical or graphical functionality of the node. A more accurate description of how each node functions and operates is beyond the scope of this article and will probably appear in future articles on the Blender Wiki at http://mediawiki.blender.org.

Compositor Node System

The Compositor Node System is an advanced post-production editor that allows for quick and easy configurable modifiers, called nodes, to be hooked together (using noodles) based on various inputs and outputs. The configurations are flexible and the results can be quite dramatic or very subtle based on design and necessity. The major feature of the Compositor Node System (Compositor) is the ability to mix various render passes (combined RGBA, Z, Vector, and Normal values) with other inputs to achieve advanced effects. These features have come about mainly through the developments made by Orange Studio and just recently by the project Plumiferos. Although designed for animations and computer-generated imagery, the Compositor has found its way into the toolbox of many graphic artists who use 3D and 2D programs to create illustrations and artwork. The possibilities are only limited by your imagination and abstraction of the node-based layout!

Inputs Values

Inputs are variable settings on all Effect and Output nodes. Input nodes do not contain inputs themselves, they use the inputs from the actual Scene. An input can either be an RGBA value (Image inputs) or a range value (Factor, Size, Value inputs). The key difference between RGBA values and range values are that each pixel represents a color/alpha value or a value representing an input amount (0-255). RGBA values are typically the pixels that make up the final image whereas range values are typically values that control the transforming of the pixels and are usually Z, Vector, Normal, Alpha or other range values.

Inputs can only accept one input channel and represent the entire picture on a per pixel basis. Inputs are either from the scene or taken from the output of another node or a set value (such as Image inputs, which can simply use an RGBA value).

An RGBA value input is always yellow and a range value input is always gray. Blue inputs are special inputs which only work with the associated scene data (Normal and Vector channels). An RGBA value input may have to be converted to be used as a range value input. An input that represents each pixel in an image is often referred to as an pixel or image map.

Output Values

Outputs values are variable settings on all Effect and Input nodes. Output nodes do not contain outputs themselves, they display the inputs. Just like Input Values, Output Values are either RGBA or range values. An RGBA value output is always yellow and a range value output is always grey and special channel values (Normal and Vector channels) are always blue. To view range values, you may have to convert it to an RGBA value or use special filters when viewing. An output that represents each pixel in an image is often referred to as a pixelmap or imagemap.

With that quick primer on inputs and outputs and what can and cannot be used, let us take a look at how these inputs and outputs can be used together with nodes. Please note that in the following explanations, an actual working model won't be described but rather how each setting can be used within each node and how, when combined with other inputs, they will change.

Input Nodes

Render Layer Nodes are inputs from the Scene. Currently the only available output settings for this node include Image (RGB), Alpha, Z (depth), Normal (mapping), and Speed (vector). The Render Layer Node contains an Image Preview and a selector for selecting Scenes and Render Layers from the selected Scene. Note that the Normal and Speed value outputs are colored blue, this is because they are special value outputs. They can be treated the same as any range value, but only work for related nodes.

The Image output displays the RGB color values for each pixel. The Alpha output is a black and white range value which represents the opacity for each pixel with a range from 0 (black/transparent) to 255 (white/solid). The Z output is a range value that gives depth information (the distance from the camera to the virtual object) on a per-pixel basis.

Image Nodes are inputs from an image file. An image file can either be a valid image type file (jpg, png, bmp, etc.) or a range image type file (exr, hdri, etc.) or even a moving image file (avi, mpeg, mov, etc.). Currently the only available output settings for this node include Image (RGB), Alpha, and Z (depth). This node also contains an Image preview.

When a file is loaded, a small icon represented by film tape appears. When you enable this, more options become available. You can set the number of frames in the animation and set the starting frame and whether or not to cycle through the frames. This makes using raw image output from the animation render easy to load into the compositor for post production, since Blender names each frame based on a number.

Also, Blender can read the file name and get the next frame of the animation. Another icon also shows up and can be enabled, which will take each frame from the file if the file is a moving image. Typically, only the Image (RGB) output is used but, if a scene file is created, you can use it as input which would deliver the Alpha and Z values to be used.

Value Node is a very simple node. You set a numeric value (0.00 to 1.00) and the value output delivers it as a range. Programmers can think of this as predefined variable.

RGB Node is basically the same thing as a Value Node except that it outputs an RGBA value instead of a range value. This node contains a color palette which makes it easy to select the desired color.

Time Node is a very practical node because it is essentially a variable Value Node. The difference is that the variation is based on the number of frames. The Factor output is a range value which is set using a curve. You can set the Start and End frame for which the curve represents. This node is useful for controlling the changes that occur during an animation, and is often used as an input for another node.

Texture Node is similar to the Image Node except that it uses internal textures from the Blender scene as the inputs. This gives you a lot of control because you can use Normal mapping to control the Offset and Scaling dynamically and can use the RGBA values as either a range Value output or an RGBA Color output. This node also includes an Image preview which displays the texture selected from the scene. This is good for adding procedural texture overlays and gradient effects.

Of all of the Input Nodes, only the Texture Input actually has input settings. These are limited to the Blender scene however, they are still dynamic. Over all, the Input Nodes are essential to using the Compositor as they serve as the base to which changes are made. They also allow for control over effects applied to other inputs. The three most commonly used Input Nodes are the Render Layer, Image, and Time nodes.

NOTE: At the time of this writing, CVS versions of Blender include new output nodes. These nodes will not be discussed in this article as they have not yet made it into the official Blender release. You should note, however, that advanced nodes do exist and will probably be making their way into future Blender versions.

Output Nodes

Compositor Node sounds kind of redundant but, it is actually the only way to output the final changes to the Render Buffer. Currently, there can only be one Compositor per scene and it only accepts Image, Alpha, and Z inputs. The node uses an Image preview for displaying the results in the Node Window. The Render Buffer (F11) will also display the Compositor node output. If no Compositor node is found the renderer will produce an error.

Viewer Node is identical to the Compositor node except that instead of displaying results to the Render Buffer, it utilizes the UV Image Editor under the Viewer Image. This is a dynamic image that displays the preview of the currently selected Viewer node. This means that you can have multiple Viewer nodes which will display the various outputs of other nodes... useful for debugging and testing different attributes of the composition.

Color Nodes

RGB Curves Node is a very powerful node because it can control the color of the image by varying the influence of each color separately, and all of the colors together. This node has a Factor Input (range value) and an Image Input (RGBA value). The Image Output is a combination of all of the RGB Curves which include Combined, Red, Green, and Blue. There are "+" and "-" symbols which zoom in and out accordingly so that you can fine tune the curve and align it more easily to the grid.

The wrench icon allows for control over the grid view and the curve handling (Auto and Vector), and it also allows you to reset the curve to the linear default. An orange grid icon allows for clipping to control the maximum values. The "X" icon deletes the currently selected point. Points are selected using the Left Mouse Button (LMB).

New points are adding by using Ctrl+LMB. This node is most commonly used for contrast control and for fine tuning the color ranges. It is easy to make very boring images interesting using this node.

Mix Node does exactly what it sounds like – it mixes two Image inputs based on a Factor and filter type and outputs the result as an RGBA value. This node has two Image inputs, but it will accept range values. This makes it good for mixing a preset value with a Time Node value to have a varying range value for nodes that don't have multiple range inputs. Among the filter types are mix, multiply, add, subtract, divide, screen, overlay, darken only, lighten only, and many others. This node is good for mixing colors as well because the Image inputs can be set as a fixed RGBA value instead of a pixel map.

Hue Saturation Node is perfect for fine tuning the RGB colors. If blue isn't blue enough, this is the node that you will find the most useful. It accepts an Image input and a Factor range value input, and outputs the modified Image. The Hue slider controls the Hue values (similar to RGB) with 0.500 being no change. The Saturation (intensity) slider controls the saturation of the Hues which can be used to flood a photo or just slightly tint an image. This node is very useful and is most popularly used as a tint filter for such night scenes where a slight blue tint is needed.

Alpha Over Node is the title effect life saver. This node allows one image to be laid over another and transparency is used as the "window" in which the background is let through. This node has two Image inputs and a Factor range value input. The Factor can be easily connected to a Time Node and then you can fade in/out the effect. Convert Premul usually blends the transparent edges together better so artifacts are removed. It also converts RGB black to Alpha, which works well for creating Alpha maps.

It's important to keep a "layer" mentality when working with this node. The top Image input is the "layer" which is placed on top of the bottom Image input and takes precedence over the pixel. Any transparent pixels then allow the background to show through. This node is great for overlays and title work as well as compositing scenes.

Z Combine Node is a very interesting node in that it can easily get tangled. It has a single Image output but accepts two Image inputs and two Z range value inputs. It combines the pixels based on the Z range values and passes that combined image to the output. The problem with this is that the Z value that corresponds to the Image (at least in the scenario where this node is useful) is the below the opposite Image input.

Thus, the noodles that connect the nodes can easily get crossed and following them gets messy if multiple nodes are used. This node is most commonly used with scene compositing. In this method, the scene is rendered in sections and layers. The layers are reassembled after post-production is completed. Usually the scene's Z channel is used to do this.

Vector Nodes

Normal Node has to be one of the coolest nodes available. I only recently figured out how amazing it really is. Basically, you can think of it as a post-production light. It uses the Normal channel as its input and then you can move a little sphere around to relocate the light source. The available outputs include the remapped Normal channel and a Dot range value output which represents the light and normal intensity in a grayscale pixel map. The possibilities for this include specular mapping, night lighting, subsurface scattering, and much more.

Vector Curves Node is similar to the RGB Curves node in that you can control the Vector channel values for the X, Y, and Z axis. The only input and output are Vector channel data which must be loaded either from a Normal node or the scene itself. It contains the standard curve widget and control features. This node is most useful for remapping the Vector channel for fine tuning motion blur, etc.

Map Value Node is another useful tool for changing value ranges. It accepts any range Value input and can limit its Minimum and Maximum values and can be further modified based on Size and Offset values. These sliders are not limited to 0.0 and 1.0 because this node is not designed only for range values of that kind. It also accepts Z, Normal, and Vector channels and can is often used in combination with the Z channel to remap it for foreground and background blur or in combination with a Vector channel to produce motion blur.

Filter Nodes

Filter Node is the most commonly used node in this set. It can be used for a broad range of effects, such as soft blurring, edge enhancements for cartoons, neon signs, decals, and much more. This node accepts an Image input which can be controlled using a Factor range value and will then modify the pixels based on a different filter or effect type. The result is passed to the Image output. The soften filter type applies a very small blur to the pixel to eliminate small artifacts. Sharpen attempts to enhance the edges between the pixels to define the edges better (this often creates artifacts). Laplace, Sobel, Prewitt, and Kirsch are all very similar and they enhance the edges and darken the rest. Shadow is kind of like engraving and applies a self shadow on the edges. This node is often used to make images that are "too perfect" seem crisper.

Blur Node is a very simple node. It takes an Image input and (based on a Size range value which influences the amount of blur) will blur the pixels based on different filters or methods. It will filter it based on an X and Y value and can even be applied to Gamma corrected values. Enabling Bokeh filtering will use a circular filter instead of the simple square method. Among the filter methods available are Flat, Quadratic, Cubic, Gaussian, CatRom, Mitch, and Tent.

Each one filters differently and any one of them can be used to obtain the desired effect. The resulting effect is applied to the image and passed to the Image output. This is very useful for Depth of Field and motion blur. It can also be used cleverly for interesting title work and overlays.

Vector Blur Node is basically a one-purpose node, but it does its job very well. It requires three inputs to generate the desired output. These three inputs are an Image, a Z channel map, and a Vector Speed input. All of these can be gathered from the scene and Vector Nodes. This method uses all three channels together to blur the image based a pixel stretching model. The inner workings of this method are beyond the scope of this article but the functionality is a very fast and efficient post-production effect. It uses a Sample value to control the quality of the blur, a MinSpeed to set the minimum stretch for every pixel and a MaxSpeed for the maximum stretch for any pixel. It then uses a BlurFactor which can be used to fine-tune the blur from strong to weak. The higher the Samples and the higher the BlurFactor, the stronger the blur. It is important to set the Min and Max speeds to inclusive enough values, but not too broad to become memory intensive. This node is most often used for motion blur.

Converter Nodes

ColorRamp Node is a dual purpose node in that it blends colors and values, and outputs both an RGB Image output and a range value Alpha output. The alpha output can be used as any value output but is most often used for transparency. The Factor input can be either a range value or a time line control to control the influence of the node. The ramp slider can use colors and transparencies to create a gradient which is useful for gradient effects on overlays and for value tuning.

RGB to BW Node is the simplest node available. It takes an RGBA Image input and outputs a range Value output. This is a grayscale representation of the RGBA Image input and can be used as such or as a value output.

Separate RGBA Node is very versatile because it allows you to modify each color channel independently. It accepts an RGBA Image input and outputs the corresponding Red, Green, Blue, and Alpha channels as grayscale values. At the time of this writing, the next official release of Blender will include a Combine RGBA Node which will do the reverse of this node.

Separate HSVA Node is essentially the same as the Separate RGBA node but it separates the Hue, Saturation, Value, and Alpha channels for the image. They are grayscale values just like the Separate RGBA node. At the time of this writing, the next official release of Blender will include a Combine HSVA Node which will do the reverse of this node.

Set Alpha Node is good for title work and overlays but has it's purpose in other nodes as well. It accepts an RGBA Image and then allows an Alpha value to be set or overwritten by a Value input. Alpha value of 0.0 is transparent and 1.0 is opaque. The combined RGBA values are passed to the Image output. This node is often used with a Time node to control the fade in or fade out of a scene.

Translate Node is the only moving animation node available at the moment (more advanced nodes for rotation, etc. are being developed). This node will translate or move an Image input a set value from the top left of the base canvas size. It will be moved X (horizontal) pixels by Y (vertical) pixels. These values can be variable X and Y inputs which is useful for animations where a Time node can be used to slide a scene or image across the canvas. The resulting Image is output which can then be placed on top of another Image for "merging down".

Group Nodes

Group nodes are nothing more than custom nodes which are made by taking the generic nodes and grouping them with Ctrl+Gkey. This group is then available as a duplicate to the current group. To ungroup the nodes, you can press Alt+Gkey. To add a group node you can either use the Add menu or press Shift+Gkey for a list of currently available nodes. Working with group nodes can save a lot of time for repetitive effects, but should be used with caution. A new group should be added, then ungrouped and regrouped as a different group node. This gets around the problem of copy nodes instead of duplicate nodes. Group nodes are most often used in the default Blender file, that way commonly used effects can be done quickly.

Conclusion

I would like to encourage you to explore the various settings and uses for the nodes. Also, visit the support forums for various reference schematics which can help you understand the mathematical and graphical functionality of each node. For practical purposes, the Compositor is a very powerful tool and shouldn't be considered a glorified Sequencer – you can do a lot more with this than you could ever do with the Sequencer. In fact, I wouldn't be a bit surprised if you never use the Sequencer again!

This has been a brief description of the Compositor, which is only half of the Node window. There is more to learn, such as the interface, the Material nodes, and the various ways in which these nodes can be used together.

Again, I encourage each user to practice using nodes and learn the "ins and outs" of this tool. Have fun!

Index Of Reflection

-by Sandra Gilbert

Index of Refraction

Refraction occurs when the energy of an incoming light wave matches the natural vibration frequency of the electrons in a material. The light wave penetrates deeply into the material, and causes small vibrations in the electrons. The electrons pass these vibrations on to the atoms in the material, and they send out light waves of the same frequency as the incoming wave. But, this all takes time.

The part of the wave inside the material slows down, while the part of the wave outside the object maintains its original frequency. This has the effect of bending the portion of the wave inside the object toward what is called the normal line, an imaginary straight line that runs perpendicular to the surface of the object. The deviation from the normal line of the light inside the object will be less than the deviation of the light before it entered the object.

The amount of bending, or angle of refraction, of the light wave depends on how much the material slows down the light. Diamonds would not be so glittery if they did not slow down incoming light much more than, say, water does. Diamonds have a higher index of refraction than water, which is to say that they slow down light to a greater degree.

One interesting note about refraction is that light of different frequencies, or energies, will bend at slightly different angles. Let's compare violet light and red light when they enter a glass prism. Because violet light has more energy, it takes longer to interact with the glass. As such, it is slowed down to a greater extent than a wave of red light, and will be bent to a greater degree. This accounts for the order of the colors that we see in a rainbow. It is also what gives a diamond the rainbow fringes that make it so pleasing to the eye.

Definition of Index of Refraction taken from
http://science.howstuffworks.com/light12.htm
For a more technical description you check out the explanation at
http://en.wikipedia.org/wiki/Refractive_index

Now why is this important? When creating materials, you need to take Index of Refraction into account. Otherwise, your materials will not be convincing. And with the new transmissivity feature added in the last release in addition to the IOR settings panel already in place, blender is capable of convincing IOR values.

Here is a list of IOR values for your reference. This image is rather small, for a clearer copy please check the pdf included with this issue.

Index of Refraction Values

Acetone	1.36	Cancrinite	1.491	Diopside	1.680	Glass, Crown	1.520	Lead	2.01	Purpurite	1.840	Strontium Titanate	2.410
Actinolite	1.618	Carbon Dioxide (gas)	1.000449	Dolomite	1.503	Glass, Crown, Zinc	1.517	Leucite	1.509	Pyrite	1.810	Styrofoam	1.595
Agalmatolite	1.550	Carbon Disulfide	1.628	Dumortierite	1.686	Glass, Flint, Dense	1.66	Magnesite	1.515	Pyrope	1.740	Sulphur	1.960
Agate	1.544	Carbon Tetrachloride	1.460	Ebonite	1.66	Glass, Flint, Heaviest	1.89	Malachite	1.655	Quartz	1.544	Synthetic Spinel	1.730
Agate, Moss	1.540	Cassiterite	1.997	Ekanite	1.600	Glass, Flint, Heavy	1.65548	Meerschaum	1.530	Quartz, Fused	1.45843	Taaffeite	1.720
Air	1.0002926	Celestite	1.622	Elaeolite	1.532	Glass, Flint, Lanthanum	1.80	Mercury (liq)	1.62	Rhodizite	1.690	Tantalite	2.240
Alcohol	1.329	Cerussite	1.804	Emerald	1.576	Glass, Flint, Light	1.58038	Methanol	1.329	Rhodochrisite	1.600	Tanzanite	1.691
Alexandrite	1.745	Ceylanite	1.770	Emerald, Synth flux	1.561	Glass, Flint, Medium	1.62725	Moldavite	1.500	Rhodonite	1.735	Teflon	1.35
Aluminum	1.44	Chalcedony	1.530	Emerald, Synth hydro	1.568	Glycerine	1.473	Moonstone, Adularia	1.525	Rock Salt	1.544	Thomsonite	1.530
Amber	1.546	Chalk	1.510	Enstatite	1.663	Gold	0.47	Moonstone, Albite	1.535	Rubber, Natural	1.5191	Tiger eye	1.544
Amblygonite	1.611	Chalybite	1.630	Epidote	1.733	Hambergite	1.559	Natrolite	1.480	Ruby	1.760	Topaz	1.620
Amethyst	1.544	Chlorine (gas)	1.000768	Ethanol	1.36	Hauynite	1.502	Nephrite	1.600	Rutile	2.62	Topaz, Blue	1.610
Anatase	2.490	Chlorine (liq)	1.385	Ethyl Alcohol	1.36	Helium	1.000036	Nitrogen (gas)	1.000297	Sanidine	1.522	Topaz, Pink	1.620
Andalusite	1.641	Chrome Green	2.4	Euclase	1.652	Hematite	2.940	Nitrogen (liq)	1.2053	Sapphire	1.760	Topaz, White	1.630
Anhydrite	1.571	Chrome Red	2.42	Fabulite	2.409	Hemimorphite	1.614	Nylon	1.53	Scapolite	1.540	Topaz, Yellow	1.620
Apatite	1.632	Chrome Yellow	2.31	Feldspar, Adventurine	1.532	Hiddenite	1.655	Obsidian	1.489	Scapolite, Yellow	1.555	Tourmaline	1.624
Apophyllite	1.536	Chromium	2.97	Feldspar, Albite	1.525	Howlite	1.586	Olivine	1.670	Scheelite	1.920	Tremolite	1.600
Aquamarine	1.577	Chrysoberyl	1.745	Feldspar, Amazonite	1.525	Hydrogen (gas)	1.000140	Onyx	1.486	Selenium, Amorphous	2.92	Tugtupite	1.496
Aragonite	1.530	Chrysocolla	1.500	Feldspar, Labradorite	1.565	Hydrogen (liq)	1.0974	Opal	1.450	Serpentine	1.560	Turpentine	1.472
Argon	1.000281	Chrysoprase	1.534	Feldspar, Microcline	1.525	Hypersthene	1.670	Oxygen (gas)	1.000276	Shell	1.530	Turquoise	1.610
Asphalt	1.635	Citrine	1.550	Feldspar, Oligoclase	1.539	Ice	1.309	Oxygen (liq)	1.221	Silicon	4.24	Ulexite	1.490
Augelite	1.574	Clinozoisite	1.724	Feldspar, Orthoclase	1.525	Idocrase	1.713	Painite	1.787	Sillimanite	1.658	Uvarovite	1.870
Axinite	1.675	Cobalt Blue	1.74	Fluoride	1.56	Iodine Crystal	3.34	Pearl	1.530	Silver	0.18	Variscite	1.550
Azurite	1.730	Cobalt Green	1.97	Fluorite	1.434	Iolite	1.548	Periclase	1.740	Sinhalite	1.699	Vivianite	1.580
Barite	1.636	Cobalt Violet	1.71	Formica	1.47	Iron	1.51	Peridot	1.654	Smaragdite	1.608	Wardite	1.590
Barytocalcite	1.684	Colemanite	1.586	Garnet, Almandine	1.760	Ivory	1.540	Periskerite	1.525	Smithsonite	1.621	Water (gas)	1.000261
Benitoite	1.757	Copper	1.10	Garnet, Almandite	1.790	Jade, Nephrite	1.610	Petalite	1.502	Sodalite	1.483	Water 100°C	1.31819
Benzene	1.501	Copper Oxide	2.705	Garnet, Andradite	1.820	Jadeite	1.665	Phenakite	1.650	Sodium Chloride	1.544	Water 20°C	1.33335
Beryl	1.577	Coral	1.486	Garnet, Demantoid	1.880	Jasper	1.540	Phosgenite	2.117	Sphalerite	2.368	Water 35°C	1.33157
Beryllonite	1.553	Cordierite	1.540	Garnet, Grossular	1.738	Jet	1.660	Plastic	1.460	Sphene	1.885	Willemite	1.690
Brazilianite	1.603	Corundum	1.766	Garnet, Hessonite	1.745	Kornerupine	1.665	Plexiglas	1.50	Spinel	1.712	Witherite	1.532
Bromine (liq)	1.661	Crocoite	2.310	Garnet, Rhodolite	1.760	Kunzite	1.655	Polystyrene	1.55	Spodumene	1.650	Wulfenite	2.300
Bronze	1.18	Crystal	2.00	Garnet, Spessartite	1.810	Kyanite	1.715	Prase	1.540	Staurolite	1.739	Zincite	2.010
Brownite	1.567	Cuprite	2.850	Gaylussite	1.517	Lapis Gem	1.500	Prasiolite	1.540	Stearite	1.539	Zircon, High	1.960
Calcite	1.486	Danburite	1.633	Glass	1.51714	Lapis Lazuli	1.61	Prehnite	1.610	Steel	2.50	Zircon, Low	1.800
Calspar	1.486	Diamond	2.417	Glass, Albite	1.4890	Lazulite	1.615	Proustite	2.790	Stichtite	1.520	Zirconia, Cubic	2.170

The last year has seen a major increase in user documentation. With the arrival of new books in several languages, downloadable manuals and class books/courses, learning Blender has become easier than ever. Let's take a look at what is currently available for our educational enjoyment.

Books

'Introducing Character Animation with Blender' by Tony Mullen, will soon be finished and is now available for pre-order at Amazon.com. It is 432 pages and is an ideal starting point for anybody interested in creating engaging, convincing character animation, giving a thorough and practical introduction to the functionality of Blender.

It comes with a companion DVD that includes the complete Blender installation executable for Windows, Mac, and Linux; the short film "Elephants Dream"; all the Blender and source files used to produce the examples and tutorials in the book; extensive links for tutorials and Blender-related resources; and other valuable Open Source software discussed in the book, including the popular BlenderPeople script.

"La 3D libre avec Blender" (French manual) by Olivier Saraja is available in France. It is around 356 pages, featuring some color plates by well known artists of the community (Stefan Zsolt, Robertt, Harkyman...).

'Blender 3D - Guia do Osorio' (Brazilian / Portuguese manual) by Allan Brito. The book is aimed both at the beginning and the advanced Blender user and it counts 448 pages.

'Blender Beginner's Bible' (Japanese manual), by Shinichi Tasaki was released in September. 'Blender Beginner's Bible' is aimed at the beginning Blender user, covering all the new features and tools that have been added to Blender since the last Japanese book was released in 2003.

Aprende en 24 Horas Blender & Yafray Diseno Grafico 3D con Software Libre (Spanish Blender 2.41 Manual)
http://www.boxel.info/morcy/index.php?entry=entry060805-185712

The printed book (or PDF version) is actually a compilation of the tutorials from the course '**Animation for Communication**' which was taught at the Escuela Superior de Informatica of Castilla-La Mancha University in Spain, in 2005/2006, by Carlos Gonzalez Morcillo

Course Work / Guided Tutorials

'Blender Basics' by Jim is the second edition of 'Blender Basics'; his classroom tutorial book. It is a very thorough, ready to use guide of 118 pages.

Neal Hirsig from Tufts University offers complete course materials for Tufts University 3D Design class can be found at http://ocw.tufts.edu/Course/28 or the Blackboard course site at: http://blackboard.tufts.edu Userword: blender Password: blender. There is a link to the course site listed under "My Courses" on the home page displayed after login. The Blackboard site is not available daily from 1:00AM - 2:00AM Eastern Standard Time.

The course contains weekly learning units consisting of over a hundred video tutorials and some 25 PDF tutorials. All associated files including completed tutorials can be downloaded. The video tutorials are in streaming Real Media format and require broadband or DSL access. The site also includes 4 course projects and examples of prior student work.

Neal invites comments on the course material and can be reached at nhirsig@tufts.edu

Official Documentation
http://mediawiki.blender.org/index.php/Main_Page
User's Manual
Game Engine Documentation
Glossary
Quick Start
Noob to Pro wikibook
Blender Summer of Documentation

Introduction to Python Scripting by Stephen Swaney (stivs)
Introduction to the Game Engine by Mal
Introduction to Character Animation by Ryan Dale
Introduction to Rigging Robert Christian (wavez)
Introduction to Physical Simulation by Felipe Bergamin Boralli
Introduction to Modeling by Michael Worcester (MickMcMack)
Introduction to Lighting by Guillermo S. Romero (gsrb3d)
Introduction to the Principles of Animation by Willian Padovani Germano (Ianwill)
Introduction to Materials and Procedural Texturing by Colin Litster (Cog)
Introduction to the Blender Database by Frédéric van der Essen (efbie)

Interested in writing articles for BlenderArt Magazine?

1. We accept the following:
   - Tutorials explaining new Blender features, 3d concepts, techniques or articles based on current theme of the magazine.
   - Reports on useful Blender events, throughout the world.
   - Cartoons related to blender world.
   - Interviews of well known Blenderheads.

2. Send submissions to sandra@blenderart.org. Send us a notification on what you want to write and we can follow up from there.

   Some guidelines you must follow.
   - Images are prefered in PNG but good quality JPG can also do. Images should be seperate from the text document.
   - Make sure that screenshots are clear and readable and the renders should be at least 800px, but not more than 1600px at maximum.
   - Sequential naming of images like image 001.png... etc.
   - Text should be in either ODT, DOC, TXT or HTML.
   - Archive them using 7zip or RAR or less preferably zip.

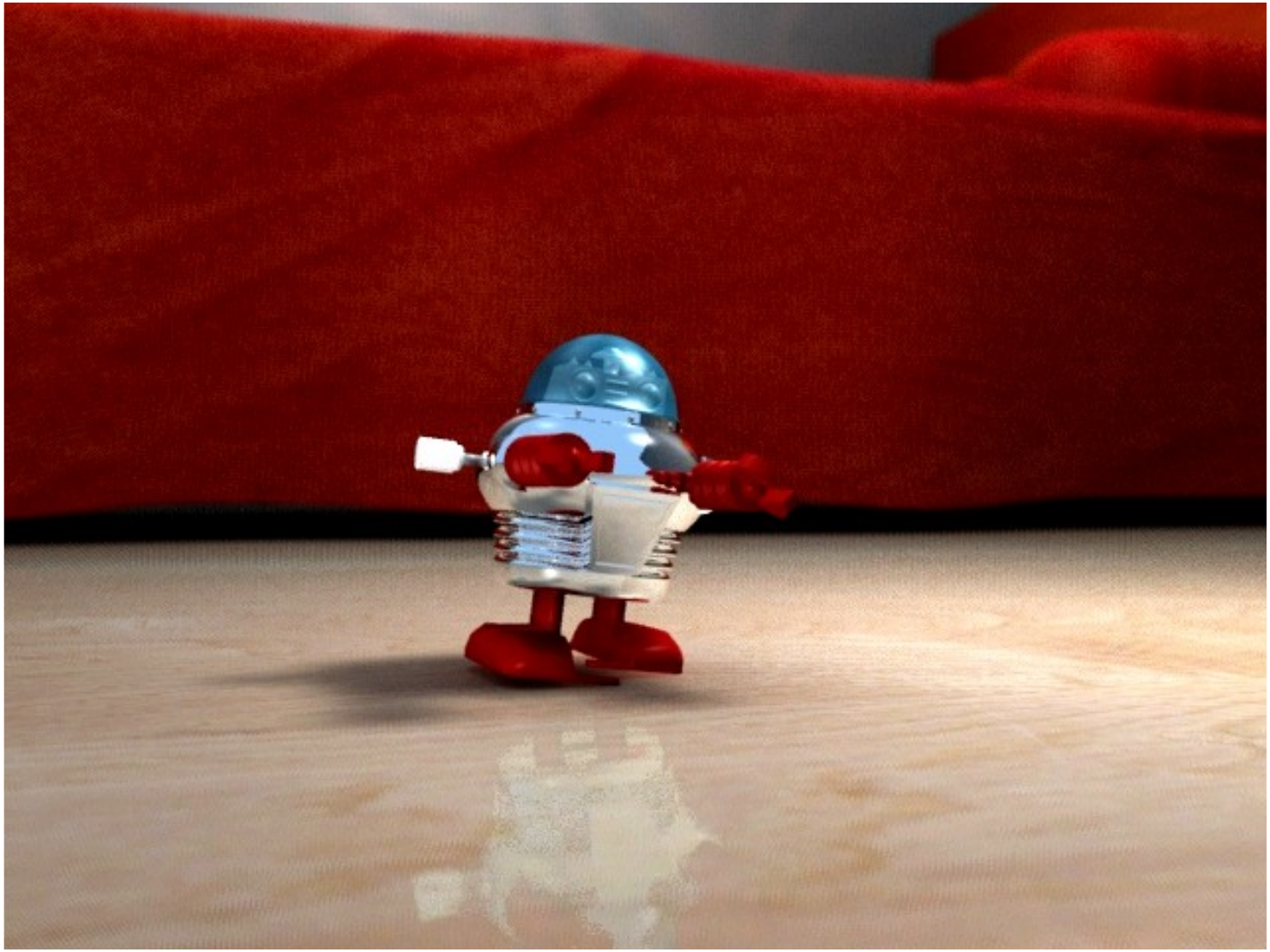
3. Please include the following in your email:
   - Name: This can be your fullname, nickname or a name of your choice.
   - Photograph: As PNG and maximum width of 256Px. (Only if submitting the article first time)
   - About yourself: Max 25 words.
   - Website: (optional)

   Note: All the approved submissions can be placed in the final issue or subsequent issue if deemed fit. All submissions can be cropped if necessary.

LaBarge - Gamer

Guilherme Lopes - Coca

Salvador Gracia Bernal - Hormijas



Scoot Morrison - Gavins Toy Robot

Zoltan Miklosi - Air Surfing

Zoltan Miklosi - Janine

Zoltan Miklosi - Jenny

George M - Coffeexp

Issue 8 January 2007

Theme: Car Modeling Mega Special!!

- Modeling Cars
- Modeling Tyres
- Car Paint and more...

Think you have some proficiency in writing articles?
Want to contribute articles and share your knowledge with blenderheads around the world?

Learn how to contribute to Blenderart Magazine here!!